TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MIKKO SIVONEN

RADIO-OHJAUKSEN INTEGROINTI VAPAASTILENTÄVÄN LENNOKIN OHJAUSJÄRJESTELMÄÄN

Diplomityö

Tarkastaja: professori Karri Palovuori
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekunnan
tiedekuntaneuvoston kokouksessa
3. joulukuuta 2014

TIIVISTELMÄ

MIKKO SIVONEN: Radio-ohjauksen integrointi vapaastilentävän lennokin ohjausjärjestelmään
Tampereen teknillinen yliopisto
Diplomityö, 59 sivua, 2 liitesivua
Joulukuu 2014
Sähkötekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Sulautetut järjestelmät
Tarkastaja: professori Karri Palovuori

Avainsanat: vapaastilentävä, lennokki, arduino

Perinteisesti vapaastilentävien lennokkien säätö tehdään lentojen välillä perustuen kokemukseen ja arvioihin sekä koelennoilla saatuihin tuloksiin. Diplomityön tavoitteena oli kehittää vapaastilentävän lennokin ohjausjärjestelmä, jota pystytään säätämään radio-ohjaimella lennon aikana. Lisäksi tarpeen oli toteuttaa tietokoneohjelma, jolla lennokin ohjaimen asetuksia ja säätöjä pystyttäisiin muuttamaan sekä lukemaan olemassa olevat säädöt ohjaimesta. Vapaastilentävä lennokki on lennokki, joka lentää ilman ulkoista ohjausta lennättäjän päästettyä sen ilmaan. Vapaastilentävillä lennokeilla kilpaillaan ympäri maailman kansallisissa ja kansainvälisissä kilpailuissa.

Työssä suunniteltiin Atmega328p-mikroprosessoriin pohjautuva lennokin ohjain, jonka ohjelmisto toteutettiin Arduino-kehitysympäristössä. Tietokoneohjelma tehtiin Java-pohjaisessa Processing-kehitysympäristössä. Elektroniikan ja ohjelmiston suunnittelussa kiinnitettiin toiminnallisuuden lisäksi erityistä huomiota virran kulutuksen optimointiin ja laitteen fyysiseen kokoon. Yhteys lennokin ohjaimesta tietokoneohjelmaan toteutettiin bluetoothin avulla ja radio-ohjaimena käytettiin radiolähetintä, joka on tarkoitettu radio-ohjattavien lennokkien lennätykseen.

Lopputuloksena syntyi toimiva ja lupaava lennokin ohjain sekä ohjelmisto. Kehitettävää jäi virrankulutuksen optimointiin ja radion kantomatkan lisäämiseen. Jatkossa tietokoneohjelma on tarkoitus muuntaa Android-alustalle sopivaksi ja mahdollisia lisälaitteita kehitetään ohjaimeen.

ABSTRACT

MIKKO SIVONEN: Integration of Radio Control with Free Flight Model Airplane Control System
Tampere University of Technology
Master of Science Thesis, 59 pages, 2 Appendix pages
December 2014
Master's Degree Programme in Electrical Engineering
Major: Embedded Systems
Examiner: Professor Karri Palovuori

Keywords: free flight, model airplane, arduino

Traditionally the adjustment of free flight model airplanes is based on experience and test flights and it is made on the ground between flights. The aim of this master's thesis was to develop and construct control system of the free flight model airplane which adjustment is possible during the flight with a radio transmitter. Developing a computer program, which allows reading and writing adjustments to model airplane control system, was an aim of the master's thesis as well. Free flight model airplane is a model airplane which is not allowed to be controlled during flight in competitions. Free flight competitions are held all over the world nationally and internationally.

Electronics was designed and constructed based on Atmega328p microcontroller. Software of the control system was implemented with Arduino integrated development environment and the computer program was made with Processing development environment. In addition to the functionality, the main criteria of the design was physical size and power consumption of the control system. Communication between computer and the control system was made using Bluetooth and a radio transmitter used was a standard radio-control model transmitter.

As a result of the master's thesis was functional and promising device and computer program. Some further development is needed with power consumption and radio range. In future the computer program will be ported to some Android based device and some optional units will be developed as well.

ALKUSANAT

Diplomityö on tehty omasta aiheestani liittyen pitkäaikaiseen harrastukseeni, lennokkeihin. Idea diplomityöstä syntyi keskusteluissa niin ikään pitkän linjan harrastajan Petri Pitkäsen kanssa. Haluan kiittää häntä ideasta ja myös diplomityön tekemisen aikana saaduista kehitysideoista.

Haluan kiittää myös professori Karri Palovuorta käytännön ratkaisuihin saamistani neuvoista liittyen elektroniikkaan ja ohjelmointiin. Lisäksi haluan kiittää Mariaa, Eilaa ja Maijaa opintojeni tukemisesta.

Tampereella 25. Toukokuuta 2015

Mikko Sivonen

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

AD	analog to digital
ADC	analog to digital converter
CRC	cyclic redundancy check
DC	direct current
EEPROM	electronically erasable programmable read only memory
ESR	ekvivalentti sarjaresistanssi
IDE	integrated development environment
ISM	industrial, scientific, medical
ISP	in-system programmer
i/o	input/output
MISO	master in slave out
MOSI	master out slave in
PWM	pulse width modulation
RISC	reduced instruction set computing
SCK	serial clock
SPI	serial peripheral interface
SRAM	static random access memory
TTY	Tampereen teknillinen yliopisto
USART	universal synchronous/asynchronous receiver/transmitter
vs.	lat. versus, vastaan

f	taajuus
C	kapasitanssi
I	virta
L	induktanssi
R	resistanssi
V	jännite

1. JOHDANTO

Vapaastilentävä lennokki on lennokki, joka lentää ilman ulkoista ohjausta lennättäjän päästettyä sen ilmaan. Vapaastilentävillä lennokeilla kilpaillaan ympäri maailman kansallisissa ja kansainvälisissä kilpailuissa. Vapaastilentävän lennokin kilpailutilanteessa ei saa käyttää radio-ohjausta, mutta kilpailujen ulkopuolella lennokkia säädettäessä se ei ole kiellettyä. Kilpailuissa keskeinen tavoite on asetetun lentoajan saavuttaminen.

Perinteisissä elektronisissa ohjaimissa lennokin ohjainpintojen liikeratoja ei pystytä muuttamaan lennon aikana vaan uudet säädöt tehdään lentojen välillä arvioiden pohjalta. Ohjausjärjestelmää, jossa lennokkia voidaan säätää lennon aikana maasta käsin ei ole markkinoilla.

Diplomityönä kehitettävä ja rakennettava ”ms-timer” on vapaastilentävän lennokin ohjausjärjestelmä, jonka avulla lennokkia voidaan säätää lennon aikana radio-ohjaimen avulla. Lennokkia ohjataan tavanomaisella lennokkiradiolähettimellä ja haluttaessa uudet liikeradat tallennetaan lennokin ohjaimen muistiin, josta ne voidaan siirtää pc-ohjelmaan ja analysoida niitä. Hyväksytyt säädöt tallennetaan ohjaimen pysyväismuistiin, jolloin ne voidaan toistaa seuraavilla lennoilla. Tarvittaessa voidaan käskyttää ohjain palauttamaan alkuperäiset säädöt myös lennon aikana.

Yhteys tietokoneen ja ms-timerin välillä tapahtuu bluetoothilla ja radio-ohjaus 435 MHz:n taajuudella. Perustana ms-timerillä on AVR:n Atmega328p-mikroprosessori ja sillä voi ohjata neljää lennokkiservoa, ulkoista led-majakkaa ja summeria. Lisäksi siinä on mahdollisuus hinaussiiman voiman mittaukseen ja siihen voi liittää I2C-väyläpohjaisia lisälaitteita.

Tässä diplomityössä keskityttiin kehittämään lennokin ohjainta käytettäväksi F1A-luokan liidokissa, mutta jatkossa ohjelmistoa voidaan kehittää soveltuvaksi myös muihin lennokkiluokkiin. F1A-luokan liidokki hinataan 50 metrin pituisella hinaussiimalla, joka kiinnitetään liidokin hinauskoukkuun. Hinausvaiheessa hinaussiima on kiinnittyneenä hinauskoukkuun ja tavoitteena on löytää nostava ilmavirtaus, johon lennokki lingotaan. Linkousvaiheessa tavoitteena on saada liidokki mahdollisimman korkealle, jopa yli 100 metriin ja sen jälkeisessä liitovaiheessa pyritään mahdollisimman pieneen vajoamisnopeuteen, eli yritetään saada lennokki lentämään vähintään kilpailukierroksen tavoiteaika.

2. VAPAASTILENTÄVÄT LENNOKIT JA NIIDEN OHJAUSJÄRJESTELMÄT

Vapaastilentävä lennokki on lennokki, joka lentää ilman lennättäjän ohjausta lennättäjän päästettyä sen ilmaan. Kilpailuissa vapaastilentävillä lennokeilla keskeinen tavoite on lentoaika. Vapaastilentävät lennokit voidaan jakaa voimanlähteen mukaan eri ryhmiin, liidokkeihin, kumimoottorilennokeihin ja poltto- tai sähkömoottorilennokeihin. Vapaastilentävien lennokkien kansainvälisiin kilpailuluokkiin kuuluu ulkolennokki-luokkien lisäksi myös sisälennokkiluokkia. [1] Tässä diplomityössä keskitytään lennokkiluokkaan F1A.

2.1 F1A-liidokin lennätys ja kilpailut

F1A-liidokki on kiinteäsiipinen lennokki, jossa ei ole ulkoista voimanlähdettä ja jonka noste syntyy aerodynaamisten voimien vaikutuksesta lennokin pintoihin. F1A-liidokin sallittu kantava pinta-ala on 32-34 dm^2 ja sen minimipaino on 410 g. Liidokki hinataan siimalla, joka saa olla enintään 50 metrin mittainen kuormitettuna 5 kg:n voimalla. Radio-ohjausta saa käyttää vain lennokin lennon keskeyttämiseen. [2]

Kansainvälisissä mestaruuskilpailuissa lennätetään 7 kilpailukierrosta, joilla jokaisella on tavoitteena saada lennokki lentämään vähintään ennalta määrätty lentoaika, esimerkiksi 3 tai 4 minuuttia. Kilpailijat, jotka ovat onnistuneet lennättämään täyden ajan kaikilla kierroksilla, jatkavat ns. ”fly-off” -kierroksille, joissa lentoaikaa pidennetään joka kierroksella. Seuraavalle kierrokselle jatkaa aina ne kilpailijat, jotka ovat lennättäneet täyden ajan ja näin jatketaan kunnes jäljellä on enää yksi kilpailija, tai jos kukaan ei ole saanut täyttä aikaa, voittaja on se jonka lennokin lentoaika on pisin.[2]

F1A-liidokin lennätys voidaan jakaa neljään osaan: hinaus, linkous, liito ja lennon keskeytys. Lennon keskeytyksestä käytetään sanontaa ”fusetus”, jonka alkuperä tulee ajalta, jolloin käytettiin lento-ajan rajoittamiseen hitaasti palavaa lankaa. Langan palaessa loppuun ohjainpinnat laukaistiin asentoon, joka pakotti lennokin hitaaseen syöksykierteeseen. Ennen sähköisiä ohjausjärjestelmiä lennokkien ohjaukseen käytettiin mekaanisia ajastimia eli ”timereita”, ja tästä syystä myös nykyisiä ohjausjärjestelmiä kutsutaan samalla nimellä.

Hinausvaiheessa hinaussiima on kiinnittyneenä liidokin hinauskoukkuun ja lennokkia voidaan lennättää siiman päässä odotellen ja tunnustellen, tavoitteena löytää nostava ilmavirtaus eli termiikki, johon lennokki lingotaan. Hinausvaiheessa lennokkia ohjataan

hinauskoukun välityksellä siten, että hinaussiimasta vedettäessä lennokki lentää suoraan ja kun siima löysätään, se alkaa kaartaa.

Sopivan hetken löydyttyä liidokki lingotaan juoksemalla ja vetämällä hinaussiimasta, jolloin hinauskoukun siiman irtoamisen estävä este aukeaa, ja kun siimasta päästetään irti, irtoaa lennokki hinaussiimasta. Siiman irtoamisen jälkeen tavoitteena on muuttaa saavutettu liike-energia potentiaalienergiaksi, eli korkeudeksi mahdollisimman tehokkaasti. Tämän aikaansaamiseksi lennokin timer ohjaa ennalta säädetyillä ohjausliikkeillä lennokin ensin nousemaan lähes pystysuoraan ylös, ja sen jälkeen oikaisee sen vaakasuoraan liitovaiheeseen. Ajanotto alkaa siitä hetkestä, kun hinaussiima irtoaa lennokista. Kuvassa 1 on tyypillinen F1A-liidokki, liidokin alla näkyvä kangassuikale on sääntöjen mukainen hinaussiiman lippu, joka auttaa ajanottajia havaitsemaan siiman irtoamisen lennokista.

Kuva 1 F1A-liidokki hinausvaiheessa

Liitovaiheessa pyritään saavuttamaan mahdollisimman pieni vajoamisnopeus ja pyritään säätämään lennokki siten, että se pysyy mahdollisessa nostavassa ilmavirtauksessa. Timer ohjaa lennokin lentämään loivaa kaartoa, jota jatketaan kunnes tavoiteltu lentoaika on saavutettu.

Fusetusvaiheessa lennokin korkeusvakaaja ohjataan noin 45 asteen kulmaan, jolloin lennokki lähtee hitaaseen syöksykierteeseen. Ilman fusetustoimintoa lennokki saattaisi lentää voimakkaassa termiikissä kymmenien kilometrien päähän ja hävitä.

F1A-liidokin lentoon voidaan vaikuttaa lennokin aerodynaamisten ohjainpintojen asennoilla. Tavallisimmin nämä ohjainpinnat ovat sivuperäsin, korkeusperäsin ja

kaarron sisäpuoleisen siiven asetuskulma. Korkeusperäsimellä ohjataan lennokin lentorataa korkeussuunnassa ja sivuperäsimellä voidaan vaikuttaa siihen, kuinka jyrkkää tai loivaa kaartoa lennokki lentää. Yleensä voidaan ohjata myös kaarron sisäpuoleisen siiven asetuskulmaa, jolloin asetuskulmaa lisäämällä saadaan hitailla nopeuksilla aikaan sisäsiiven jarrutusvaikutus ja nopeilla nopeuksilla lennokin pituusakselin suuntainen kallistuminen johtuen siiven kasvaneesta nostovoimasta. Ohjainpintojen liikuttaminen tapahtuu radio-ohjatuissa lennokeissa yleisesti käytettävillä servoilla.

Liidokin hinauskoukku on saranoitu yläpäästään jolloin hinaussiimaa vedettäessä koukku liikkuu eteen ja löysättäessä taakse. Hinaussiiman irtoamisen estävä este voi olla toimintaperiaatteelta mekaaninen jousikuormaan perustuva tai servolla toimiva. Mekaanisesti toimivan koukun este aukeaa jousikuorman ylittäessä ennalta säädetyn rajan, kun taas servolla toimivaa estettä voidaan ohjata auki tai kiinni ohjelmallisesti. Servotoiminen hinauskoukku vaatii käytännössä myös hinaussiiman koukkuun kohdistaman voiman mittauksen, jotta voidaan asettaa ehdot esteen avaamiselle ja sulkemiselle. Hinauskoukun ja hinaussiiman esteen tilatietojen ja mahdollisen hinaussiiman vetovoiman mittauksen perusteella ohjataan lennokin ohjainpintoja hinausvaiheessa ja päätellään milloin siirrytään linkous- tai liitovaiheeseen.

2.2 Perinteiset elektroniset timerit ja ms-timer

Perinteisten elektronisten timereiden kanssa lennokkien säätö tapahtuu käytännössä siten, että uudelle lennokille arvioidaan ensin sopivat säädöt ja asetukset. Lennokkia lennätetään ja lennon jälkeen muutetaan säätöjä oletetusti haluttuun suuntaan, jonka jälkeen uusia säätöjä testataan taas lennättämällä. Jotta lennokki saadaan lentämään halutulla tavalla, koelentoja tarvitaan usein vähintään kymmeniä. Diplomityössä kehitettävä ohjain ”ms-timer” mahdollistaa lennokin säätöjen muuttamisen lennon aikana käyttäen radio-ohjattavien lennokkien lennätykseen tarkoitettua radio-ohjainta. Kun kaikki lennon vaiheet voidaan säätää lennon aikana, periaatteessa säätämiseen riittää vain yksi koelento.

Radio-ohjaimella tapahtuva säätö täytyy pystyä kytkemään päälle ja pois lennon aikana. Lisäksi haluttiin toiminto, jolla alkuperäiset ohjaimen pysyväismuistiin tallennetut arvot voidaan palauttaa kesken lennon. Tätä tarkoitusta varten käytetään radio-ohjaimen 3-asentoista kytkintä, jonka asennot ovat radiosäätö päälle, radiosäätö pois ja alkuperäisten asetusten palautus.

Monissa elektronisissa timereissa yhteys tietokoneeseen tai ohjelmointilaitteeseen muodostetaan langallisesti, mutta koska kokemus on osoittanut, että lennätys-olosuhteissa johdot ja liittimet sekä vesi, jää ja hiekka yhdessä johtavat usein ongelmiin, ms-timerissä yhteys tietokoneohjelmaan muodostetaan bluetoothin välityksellä.

Ms-timerissä on myös mahdollisuus liittää siihen ulkoinen summeri ja lennokin rungon ulkopintaan asennettava led-majakka. Näillä voidaan viestittää lennättäjälle, missä tilassa tai lennon tilanteessa ohjain on. Voi olla esimerkiksi hyödyllistä antaa äänimerkki kun hinauskoukun este aukeaa tai varoittaa äänimerkillä alhaisesta akkujännitteestä. Led-majakan vilkutus lennon aikana myös parantaa lennokin näkyvyyttä ja helpottaa ajanottamista. Servoja tarvitaan F1A liidokkiin yleensä kolme tai neljä, korkeusperäsimelle, sivuperäsimelle, sisäsiivelle ja mahdollisesti koukkuun hinaussiiman esteelle.

Useimpiin perinteisiin lennokin ohjaimiin voidaan kytkeä radiofuse-laitteen vastaanotin, jonka avulla lennokin ohjain voidaan ohjata missä tahansa vaiheessa fusetustilaan. Ms-timerissä radiofuse-toiminto on integroituna eikä erillistä laitetta tarvita. Lisälaitteita ms-timeriin voi kytkeä siihen suunniteltuun I2C-väyläliittimeen, josta lisälaite voi ottaa myös käyttöjännitteensä. Teholähteenä ms-timerissä käytetään kaksikennoista litiumpolymeeriakkua.

3. ELEKTRONIIKKA

F1A-liidokin ja ylipäätään vapaastilentävän lennokin paino ja ilmanvastus on sen suorituskyvylle kriittisiä ominaisuuksia. Tästä johtuen lennokkien rungot muotoillaan mahdollisimman kapeiksi ja sulavalinjaisiksi, joten rungon sisällä tila on yleensä hyvin rajallinen. Liidokin rungossa hinauskoukku ja servot on käytännössä asennettava peräkkäin rungon pituusakselin suuntaisesti korkeussuunnassa rungon keskelle. Akku sijoitetaan useimmiten rungossa aivan eteen, jotta painopiste saadaan säädettyä kohdalleen. Tyhjää tilaa jää rungon yläosaan, jonne timerin piirilevy on paras sijoittaa. Piirilevystä suunniteltiin muodoltaan pitkä ja kapea, jotta sen sijoitus rungon tyhjään yläosaan onnistuisi helposti.

3.1 Yleiset vaatimukset piirilevylle

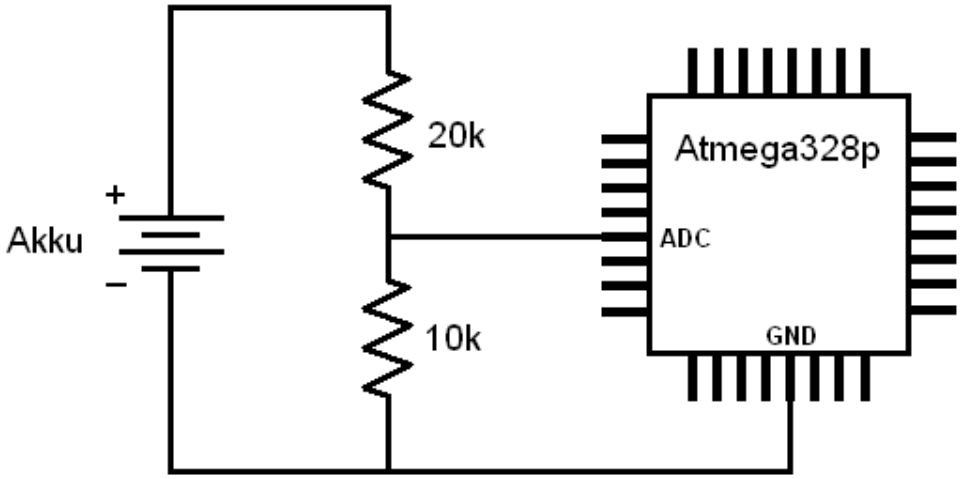
Reititykset piirilevyllä tehtiin siten, että suurivirtaisten johtimien leveydet pyrittiin maksimoimaan ja pienivirtaisten johtimien reitityksetkin yritettiin pitää vähintään 0,3 mm:n levyisinä. Ledien etuvastukset valittiin pitäen silmällä tehon kulutuksen optimointia. Etuvastuksiksi valittiin 2 kΩ:n vastukset, jolloin käyttöjännitteen ollessa 3,3 V ja ledin kynnysjännitteen ollessa 1,8 V, saadaan Ohmin lain mukaisesti ledin virraksi 0,75 mA. Passiivikomponentteina käytettiin pääasiassa 0603-koteloisia komponentteja. Piirilevysuunnittelu tehtiin Cadsoft Eagle PCB design -ohjelman ilmaisversiolla. Lennokin ohjaimen kytkentäkaavio kokonaisuudessaan on esitetty liitteessä A.

Mikroprosessorin ohjelmointia varten täytyi piirilevylle suunnitella ohjelmointiliitin ISP-väylälle (In-System Programming). ISP-väylä on Atmel:n AVR-mikrokontrollereiden SPI-väylään (Serial Peripheral Interface) perustuva ohjelmointiväylä. ISP-väylän kautta voidaan ohjelmointilaitteen avulla kirjoittaa mikrokontrollerin flash- ja EEPROM-muistia ohjelmointilaitteen avulla. SPI-väylä sisältää kolme napaa, MISO (Master In Slave Out), MOSI (Master Out Slave In) ja kellolinjan SCK (Serial Clock). Jotta AVR-mikrokontrolleri saadaan ohjelmointitilaan, täytyy sen reset-linja saada aktiiviseksi, joten ISP-väylä sisältää myös reset-linjan. Ohjelmointilaitteen ja mikrokontrollerin maatasojen on oltavat samat, mistä syystä ISP-liittimessä on maadoitusnapa (GND). Ohjelmoitava laite voi ottaa käyttöjännitteensä ohjelmoinnin aikana ohjelmointilaitteesta, joten standardin mukaisessa ISP-liittimessä on käyttöjännitenapa (VCC). [3] Tilan säästämiseksi ms-timerin ohjelmointiliittimestä jätettiin pois käyttöjännitenapa, joten ohjelmoitaessa se tarvitsee ulkoisen virtalähteen.

Mahdollisia jatkossa kehitettäviä lisälaitteita ja -ominaisuuksia varten ms-timerin piirilevyllä on liitin myös I2C-väylälle. I2C-väylä on Philipsin kehittämä master/slave

periaatteella toimiva kaksisuuntainen sarjaliikenneväylä. [4] Ms-timerin I2C-liitynnässä on väylän data- ja kellolinjojen lisäksi käyttöjännite- ja maaliittimet lisälaitteen jännitteen syöttöä varten.

Akkukäyttöisessä laitteessa taloudellisuus tehon käytössä on tärkeää. Parhaan mahdollisen hyötysuhteen saavuttamiseksi käyttöjännitteiden regulointia varten ms-timeriin suunniteltiin hakkurityyppiset regulaattorit. Jotta akkujännitettä voitaisiin valvoa ja tarvittaessa estää akun tyhjentyminen, suunniteltiin piirilevylle akkujännitteen mittauskytkentä. Jännitettä mitataan prosessorin analogiatulon avulla, ja jotta mittaus onnistuu, täytyy mitattava jännite laskea alle mikrokontrollerin käyttöjännitteen. Tämä toteutettiin vastusjaon avulla. Datalehden mukaan Atmega328p:n analogiatulot on optimoitu korkeintaan 10 kΩ:n mitattavan lähteen ulostuloimpedanssille, joten vastuksien arvoiksi valittiin 10 kΩ ja 20 kΩ. Ohmin lain mukaisesti prosessorin pinnille johdettu jännite kuvan 2 mukaisella kytkennällä on 1/3 akun jännitteestä, jolloin akkujännitteen ollessa 8,4 V prosessorin ADC-pinnillä on n. 2,8 V:n jännite.



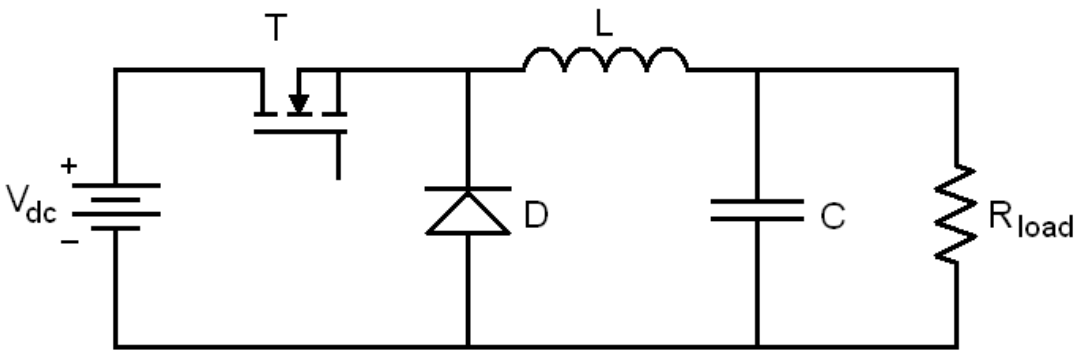
Kuva 2 Akkujännitteen mittauskytkennän periaatekuva

3.2 Käyttöjännitteet

Teholähteenä ms-timerissä käytetään erillistä kaksikennoista litiumpolymeeriakkua (LiPo), jonka kennojännite on 3,7 V. Täyteen ladatun LiPo-akun kennojännite on 4,2 V ja turvallisena kennojännitteen alarajana pidetään n. 3,6 – 3,8 V. Kaksikennoisen akun napajännite voi siis vaihdella n. 7,2 – 8,4 V:n välillä.

Käytetyn bluetooth-moduulin (HC-06) ja UHF-radiomoduulin (RFM22B) nimellinen käyttöjännite on 3,3 V, joten se valittiin myös mikroprosessorin käyttöjännitteeksi. Lennokkiservojen käyttöjännitteet ovat tyypillisesti välillä 4,8 V – 6 V, joten niille suunniteltiin oma 5 V:n jännitteen syöttö.

Parhaan hyötysuhteen saavuttamiseksi käyttöjännitteiden regulointi tehtiin step-down -tyyppisillä DC-DC jännitehakkureilla. Step-down hakkuri on jännitettä laskeva teholähde. Se koostuu puolijohdekytkimestä (T), diodista (D), kelasta (L) ja kapasitanssista (C). Kytkintransistorin johtaessa kelan virta kasvaa ja energiaa latautuu kelan magneettikenttään. Kytkimen avautuessa kela pyrkii ylläpitämään virtaansa ja diodi alkaa johtamaan. Magneettikentän energiaa puretaan kondensaattoriin ja kuormaan. Kun kytkintä avataan ja suljetaan riittävällä taajuudella hakkurin ulostulojännitteeksi muodostuu lähes tasajännite. Jännite aaltoilee hieman johtuen kelan virran lineaarisesta noususta ja laskusta. Mitä suurempi on kytkentätaajuus, sitä pienempi on hakkurin lähtöjännitteen jännitevaihtelu, eli jänniterippeli. Lähtöjännitteen tasajännitekomponentin suuruus riippuu pulssisuhteesta, millä tarkoitetaan kytkimen johtamisaikaa suhteessa koko kytkentäjakson aikaan. Mitä suurempi pulssisuhde on, sitä suurempi on hakkurin ulostulon DC-jännite. Step-down hakkurin hyötysuhde voi olla n. 95%. [5] Kuvasta 3 nähdään step-down hakkurin piirikaavion periaatekuva.



Kuva 3 Step-down hakkurin periaatekuva

Hakkuripiiriksi 3,3V:n käyttöjännitteen syöttöön valittiin Analog Device:n ADP2301, joka on 1,4 MHz:n vakiokytkentätaajuudella toimiva step-down regulointipiiri, ja johon on integroitu mosfet-tyyppinen tehotransistori. Piirin nimellisvirta on 1,2 A. Piiristä on olemassa myös 700 kHz:n kytkentätaajuudella toimiva ADP2300 ja valintaan näiden kahden välillä vaikuttaa se, onko parempi hyötysuhde vai ratkaisun vaatima pinta-ala tärkeämpi kriteeri. [6] Hakkuripiirien valintaa ohjasi nimenomaan hyvin rajallinen käytettävissä oleva tila, joten tästä syystä valittiin 1,4MHz:n kytkentätaajuudella toimiva ADP2301.

Hakkurikelan mitoitukseen valmistaja antaa piirin datalehdessä ohjeeksi mitoittaa peak-to-peak virtarippeli 30 % maksimi kuormavirrasta parhaan transienttivasteen ja hyötysuhteen saavuttamiseksi. Tällä periaatteella laskettu induktanssin arvo saadaan laskettua kaavalla

$$L = \frac{(V_{IN} - V_{OUT})}{0.3 \cdot I_{LOAD} \cdot f_{sw}} \cdot \left(\frac{V_{OUT} + V_D}{V_{IN} + V_D}\right), \quad (1)$$

missä f_{sw} on kytkentätaajuus, L hakkurikelan induktanssi, V_D diodin myötäsuuntainen kynnysjännite, V_{IN} sisääntulojännite ja V_{OUT} ulostulojännite.

Maksimikuormavirtaa arvioitaessa täytyi ottaa huomioon myös mahdollisesti jatkossa I2C-väylään liitettävien laitteiden virran tarve. Diodi valittiin hakkupiirin datalehden suosituksen sekä saatavuuden ja mahdollisimman pienen koon perusteella. Valitun schottky-diodin myötäsuuntainen kynnysjännite on 0,5V. Kun maksimivirta oletettiin 500mA:n suuruiseksi, hakkurikelan induktanssin arvoksi saatiin 9,2 - 10,3 µH riippuen syöttöjännitteestä, joka voi vaihdella välillä 7,2 - 8,4 V. Hakkurikelan arvoksi valittiin saatujen arvojen perusteella 10 µH.

Sisääntulon kondensaattorin täytyy kestää maksimisyöttöjännite ja maksimisyöttövirran vaihtelu. Syöttökondensaattoriksi suositellaan datalehdessä X5R- tai X7R-sarjan keraamikondensaattoria. Datalehden mukaan 10 µF on riittävä useimpiin sovelluksiin. Kondensaattoriksi valittiin datalehden suosituksen mukainen tyyppi.

Ulostulon kondensaattorit vaikuttavat sekä ulostulon jänniterippeliin että regulaattorin dynamiikkaan. ADP2301 on suunniteltu toimimaan pienillä keraamikondensaattoreilla, joilla on pieni ekvivalentti sarjaresistanssi ja –induktanssi. [6]

Datalehden mukaan jänniterippeli voidaan laskea kaavalla

$$\Delta V_{ripple} = \Delta I_{ripple} \cdot \left(\frac{1}{8 \cdot f_{sw} \cdot C_{OUT}} + ESR_{Cout} \right), \quad (2)$$

missä ΔV_{ripple} on ulostulon jänniterippeli, ΔI_{ripple} hakkurikelan virtarippeli, f_{sw} kytkentätaajuus, C_{OUT} ulostulon kondensaattorin kapasitanssi ja ESR_{Cout} ulostulon kondensaattorin ekvivalentti sarjaresistanssi.

Kelan virtarippeli voidaan laskea kaavalla

$$\Delta I_{ripple} = \frac{(V_{IN} - V_{OUT})}{L \cdot f_{sw}} \cdot \left(\frac{V_{OUT} + V_D}{V_{IN} + V_D} \right), \quad (3)$$

missä ΔI_{ripple} on hakkurikelan virtarippeli, V_D diodin myötäsuuntainen kynnysjännite, V_{IN} hakkurin sisääntulojännite, V_{OUT} hakkurin ulostulojännite, L hakkurikelan induktanssi ja f_{sw} kytkentätaajuus.

Datalehden suositusten mukaisesti ulostuloon valittiin Muratan valmistama 22 µF:n keraamikondensaattori, jonka ekvivalentin sarjaresistanssin arvo on valmistajan mukaan 1,4Mhz:n taajuudella n. 0,004 Ω. Kaavoilla 2 ja 3 laskettuna virtarippeliksi saadaan 137,48 -155,54 mA ja jänniterippeliksi 1,1 - 1,2 mV syöttöjännitteen ollessa välillä 7,2 – 8,4V.

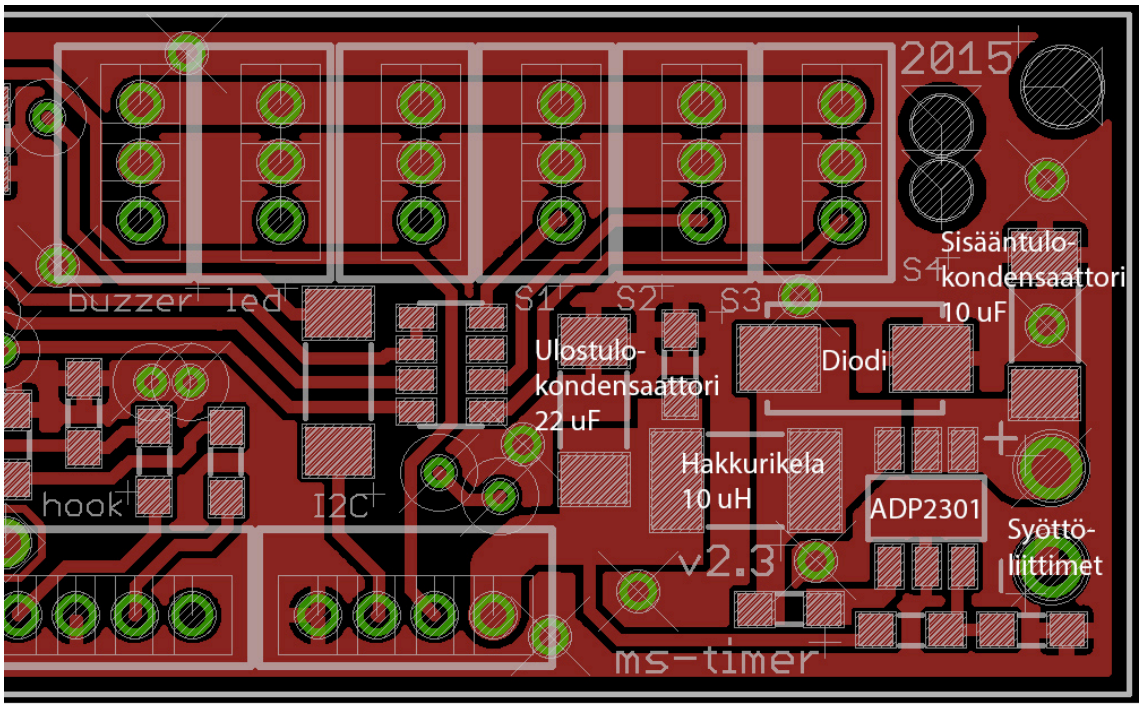
Servojen virransyöttöä varten 5 V:n käyttöjännitteen regulointiin valittiin Analog Devicen ADP2303 step-down hakkuripiiri, jossa myös on sisäinen mosfet-transistori. Piirin kytkentätaajuus on 700kHz ja sen nimellisvirta on 3A. Piiristä on olemassa myös ADP2302 versio, jonka nimellisvirta on 2A.[7]

F1A-liidokissa käytettyjen pienikokoisten servojen maksimivirta on n. 500 mA. Periaatteessa piirilevyn suunnittelu mahdollistaa kuuden servon kytkemisen timeriin, mutta on epätodennäköistä, että kaikki servot ottavat samaan aikaan maksimivirran. Hakkuripiiriksi valittiin 3 A:n nimellisvirtainen ADP2303, mutta oheiskomponenttien mitoituksessa käytettiin 2 A:n maksimivirtaa. Myös 5 V:n hakkurin diodi valikoitui kotelon koon ja saatavuuden perusteella. Valitun shottky-diodin myötäsuuntainen kynnysjännite on 0,45 V ja kelan mitoituksessa käytettiin samaa 30 %:n rippelivirran mukaista laskukaavaa 1. Syöttöjännitteen ollessa 7,2 – 8,4 V saatiin kelan induktanssiksi 3,7 – 5,0 µH. Tulosten perusteella valittiin hakkurikelan induktanssin arvoksi 4,7 µH.

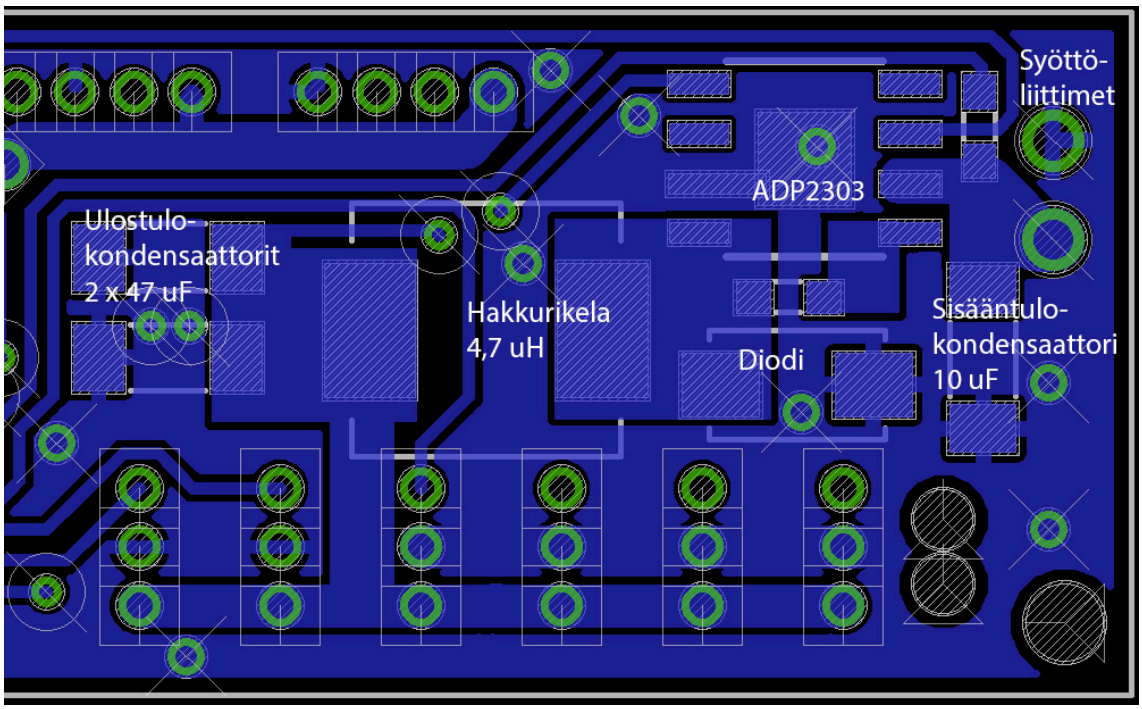
Sisääntulokondensaattoriksi valittiin datalehden suosituksen mukainen 10 µF:n kondensaattori. Ulostulon kondensaattoreiksi valittiin datalehden suosituksia mukaillen 2 x 47 µF: kondensaattorit, joiden ekvivalentin sarjaresistanssin arvo 700 kHz:n taajuudella on valmistajan mukaan n. 0,002 Ω. Kaavoilla 2 ja 3 saadaan virtarippeliksi laskettua 477,64 - 638,64 mA ja jänniterippeliksi 1,9 – 2,5 mV jännitteen ollessa 7,2 – 8,4 V.

Piirilevyn layout-suunnittelu on hyvin tärkeässä osassa hakkuriteholähteen parhaan suorituskyvyn saavuttamiseksi. Huono suunnittelu voi vaikuttaa regulaattorin vakauteen sekä sähkömagneettiseen yhteensopivuuteen. Kondensaattorit, kela ja diodit tulee sijoittaa mahdollisimman lähelle hakkuripiiriä, käyttäen mahdollisimman lyhyitä reititystä. Suurivirtaiset reititykset täytyy pitää mahdollisimman lyhyinä, mutta mahdollisimman leveinä. Suurivirtaiset reititykset ovat hakkupiirin syöttö ja syöttökondensaattori, hakkuripiirin ulostulosta kelan ja ulostulokondensaattorin kautta maahan sekä maasta diodin läpi kelalle. Maadoitukset täytyy varmistaa mahdollisimman monella läpiviennillä. Takaisinkytkentä tulee tehdä mahdollisimman lyhyellä vedolla hakkuripiirille. [7] Kuvissa 4 ja 5 on hakkuriteholähteiden piirilevy-layoutit piirilevysuunnitteluohjelman näkymästä. Kuviin on lisätty oleellisten komponenttien nimet.

Hakkuriteholähteiden haittapuolena on niiden tuottamat häiriöt, jotka saattavat vaikuttaa niiden syöttämien ja lähellä olevien laitteiden toimintaan ja saattavat esimerkiksi aiheuttaa radio-ohjattavaan laitteeseen radiohäiriöitä. Hakkuriteholähde säteilee ympäristöönsä perustaajuisen sähkömagneettisen säteilyn lisäksi laajan sarjan harmonisia taajuuksia, jotka kuitenkin vaimenevat nopeasti taajuuden kasvaessa. Koska käytettyjen hakkuriteholähteiden ja radio- sekä bluetooth-laitteiden taajuuserot ovat varsin suuret, oletettiin, ettei ongelmia kohdata. Lisäksi käytetty taajuushyppelytekniikka parantaa radiohäiriöiden sietokykyä, jos radiotaajuus osuu jonkin n-kertaluvun harmoniselle taajuudelle, seuraavan taajuushypyn jälkeinen radiotaajuus ei todennäköisesti enää osu.



Kuva 4 3,3V:n step-down hakkurin piirilevy-layout Eagle suunnitteluohjelmassa



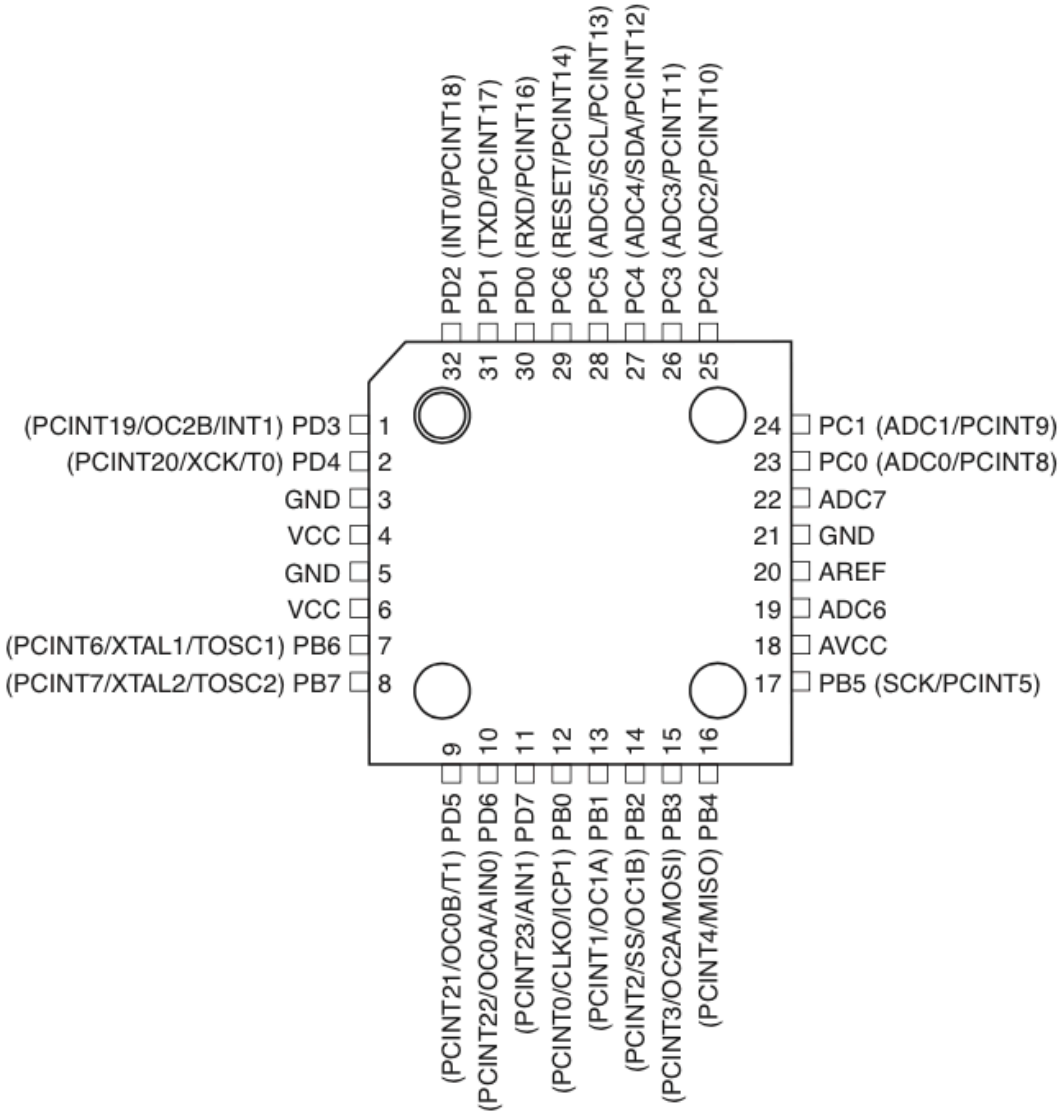
Kuva 5 5V:n step-down hakkurin piirilevy-layout Eagle suunnitteluohjelmassa

3.3 Mikroprosessori

Mikroprosessoriksi valittiin Atmega328p, joka on yksi yleisimmistä Atmel:n mikroprosessoreista. Sitä käytetään useassa OpenLRS-laitteessa sekä monessa Arduino-laitteessa, joihin ms-timerin toteutus pohjautuu.

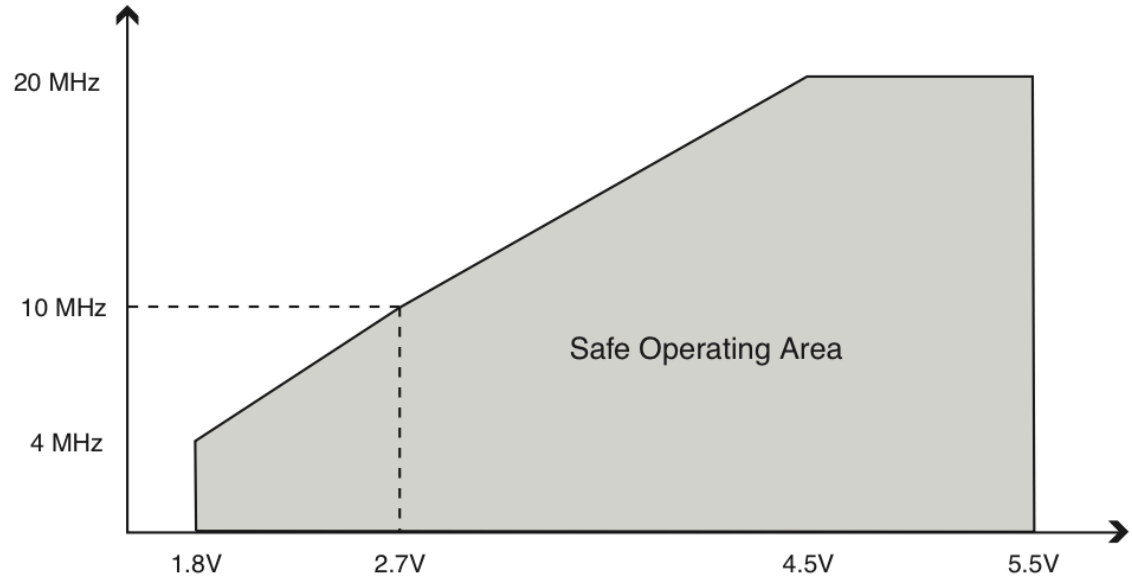
Atmega328p on Atmegan AVR-perheeseen kuuluva RISC-arkkitehtuuriin (Reduced Insruction Set Computing) pohjautuva 8-bittinen mikro-ohjain, jossa on 32 kilotavun flash-ohjelmamuisti, 2 kilotavun SRAM-muisti ja 1 kilotavun EEPROM-muisti. Siinä on 23 yleiskäyttöistä i/o-pinniä, 8-kanavainen 10-bittinen AD-muunnin ja sen maksimikellotaajuus on 20 MHz. Muita ominaisuuksia ovat mm. 3 timer/counter:ia, 6 PWM-kanavaa, USART-sarjaliikenneväylä (Universal Synchronous/Asynchronous Receiver/ Transmitter) sekä SPI- ja I2C-väylät. [8]

Atmega328p-mikroprosessoria saadaan useassa eri kotelossa kuten esimerkiksi 28-napaisessa PDIP- ja 32-napaisessa TQFP-kotelossa. Fyysisten erojen lisäksi 28- ja 32-napaisten mikroprosessorien välillä eroa on ADC-kanavien määrässä, 32-napaisissa prosessoreissa on 8 ADC kanavaa, kun 28-napaisissa on kaksi vähemmän. [8] Kuvassa 6 on Atmega328p-mikroprosessorin napajärjestys 32-napaisessa TQFP-kotelossa.

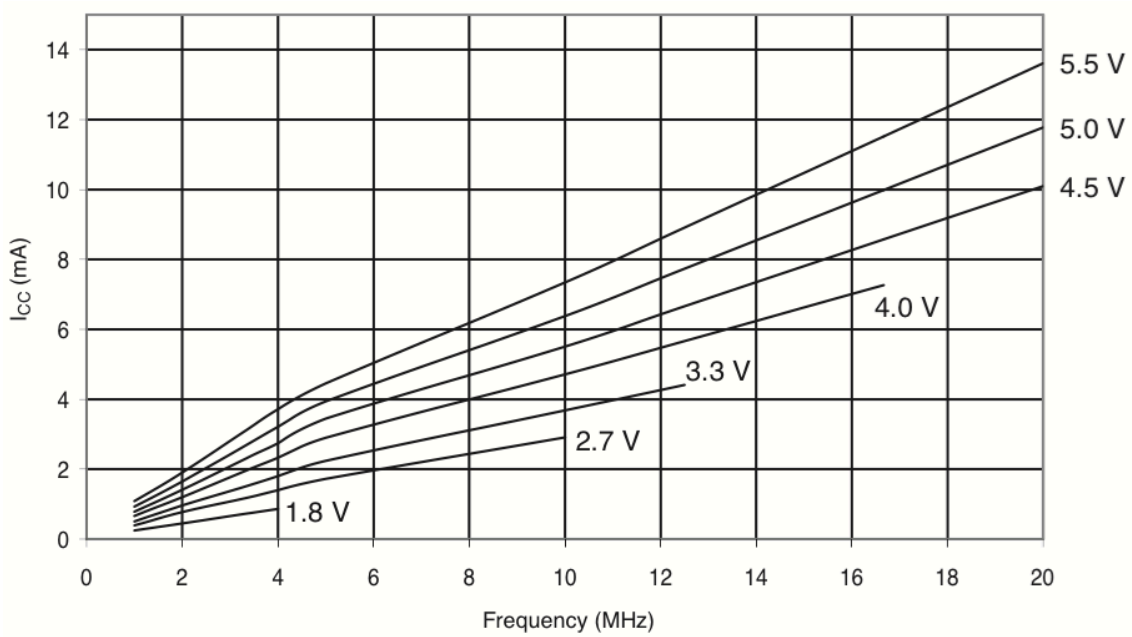


Kuva 6 Datalehden kuva Atmega328p-mikroprosessorin pinnijärjestystä 32-napaisessa TQFP-kotelossa

Mikroprosessorin suositeltu maksimikellotaajuus on riippuvainen käyttöjännitteestä. Datalehdessä esitetty maksimikellotaajuus vs. käyttöjännite -kuvaaja on esitetty kuvassa 7. Siitä voidaan arvioida, että 3,3 V:n käyttöjännitteellä maksimikellotaajuus on n. 12 MHz. Kellotaajuudeksi valittiin 8 MHz ja kellopulssin lähteeksi ulkoinen kide sisäisillä kondensaattoreilla varustettuna.



Kuva 7 Atmega328p-mikroprosessorin maksimikellotaajuus vs. käyttöjännite

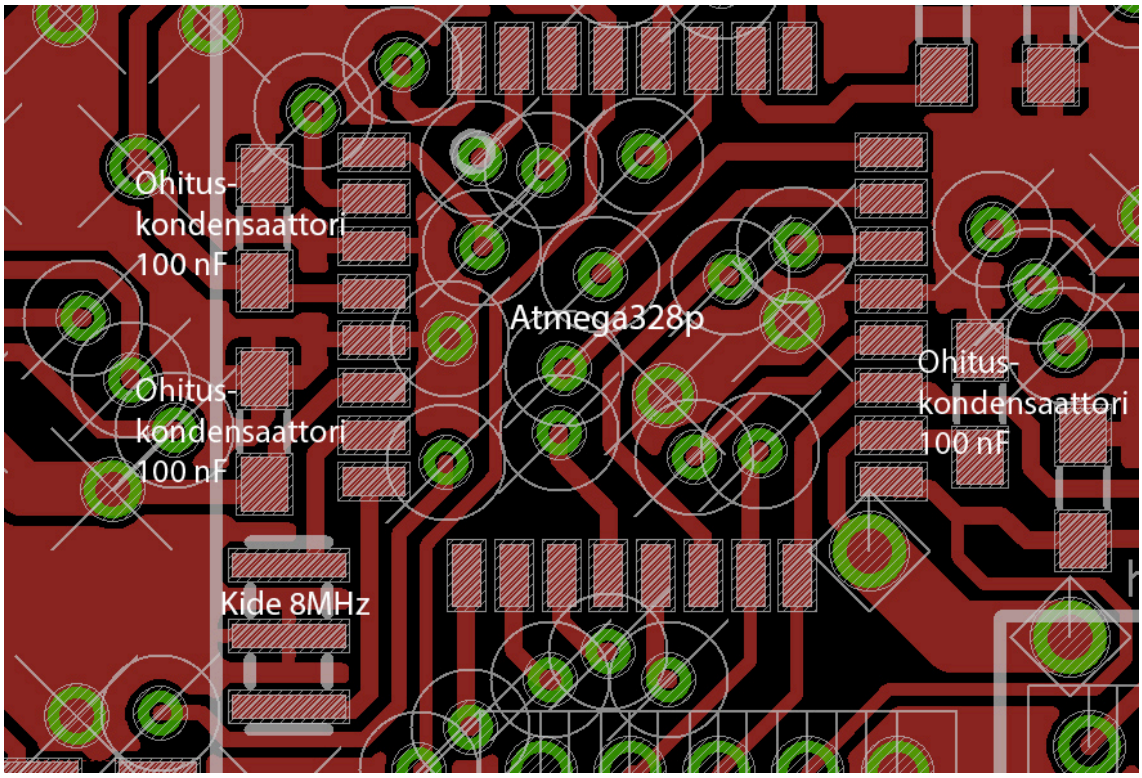


Kuva 8 Atmega328p:n keskimääräinen virrankulutus aktiivisessa tilassa eri käyttöjännitteillä

Käyttöjännite ja kellotaajuus vaikuttavat myös mikroprosessorin virrankulutukseen, mitä korkeampi on käyttöjännite ja kellotaajuus, sitä suurempi on mikroprosessorin

ottama virta. Kuvassa 8 on Atmega328p:n datalehdellä esitetty kuvaaja mikroprosessorin ottamasta keskimääräisestä virrasta sen ollessa aktiivinen. Kuvasta nähdään, että 3,3 V:n käyttöjännitteellä ja 8 MHz:n kellotaajuudella virta on n. 3 mA, kun vertailun vuoksi esimerkiksi 5 V:n käyttöjännitteellä ja 16 MHz:n kellotaajuudella vastaava virta on n. 9,5 mA.

Mikroprosessorin tasaisen virran syötön takaamiseksi suunniteltiin sen käyttöjännitteen syöttönapojen läheisyyteen ns. ohituskondensaattorit. Ohituskondensaattoreiden tehtävänä yleisesti on mm. estää muiden piirien aiheuttamien suuritaajuisten jännitevaihteluiden pääseminen suojattavan piirin käyttöjännitenapoihin, sekä estää piirin omien häiriöiden pääsy muualle piirilevylle. Kuvassa 9 on esitetty mikroprosessorin ohituskondensaattoreiden sijoitukset Eagle:n näkymässä.



Kuva 9 Mikroprosessorin ohituskondensaattorit Eagle –suunnitteluohjelman näkymässä

3.4 Servot, led-majakka ja summeri

Lennokin ohjaimella täytyy pystyä ohjaamaan neljää servoa ja lisäksi ulkoista led-majakkaa sekä summeria. Led-majakkaa käytetään lennokin näkymisen parantamiseksi ja lisäksi sillä voi informoida lennättäjää ohjaimen ja lennokin tilasta. Myös summeria käytetään ohjaimen tilan viestittämiseen sekä varoitusäänien kuten alhaisen akkujännitteen varoitusäänen tuottamiseen.

Lennokin ohjainpintoja liikutetaan lennokkiservoilla. Servojen liittämiseksi ohjaimeen täytyi piirilevylle valita sopivat liittimet. Servoille tuodaan käyttöjännite ja jokaiselle servolle oma ohjauspulssi mikrokontrollerin i/o-pinniltä. Led-majakan ja summerin liitännät suunniteltiin siten, että ne voidaan tarvittaessa muuttaa ohjelmallisesti servoliitännöiksi. Liittimiksi valittiin JST zhr -tyyppinen liitin (kuva 10) lähinnä yhteensopivuussyistä, sillä joissain olemassa olevissa lennokin ohjaimissa on käytetty samaa liitintä servojen liittämiseen. Lisäksi se on pienikokoinen ja käytännössä havaittu luotettavaksi. Ohjaussignaalit johdettiin mikroprosessorille 1 kΩ:n virranrajoitus-vastuksien kautta.

Kuva 10 Servoliittimenä käytetty JST zhr-tyyppinen liitin

Servoja ohjataan ohjauspulssilla, jonka pituus voi vaihdella n. 1 - 2 ms:n välillä, 1,5 ms:n ohjauspulssin tuottaessa servon neutraalin asennon. Servon tyypistä riippuen ääriasentoihin tarvittava pulssinpituus voi olla hieman yli 2 ms tai alle 1 ms, mutta varsinaista standardia ei ohjauspulssista ole olemassa. Ohjauspulssia täytyy toistaa n. 20 ms:n välein, joten laskennallisesti ohjaukselle saadaan n. 50 Hz:n resoluutio.

3.5 Liidokin hinauskoukun tilatiedot ja hinaussiiman voiman mittaus

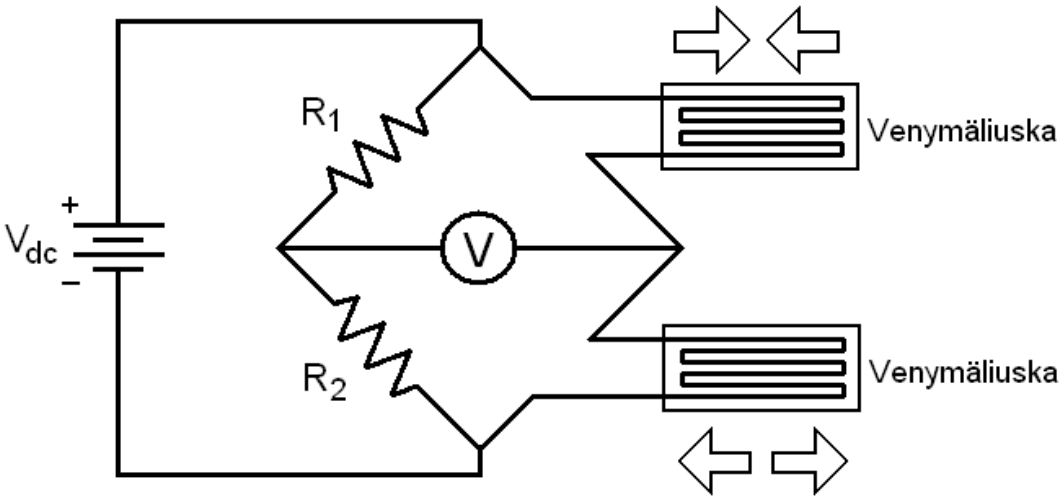
Hinauskoukun tilatietojen välittämiseksi mikrokontrollerille ohjaimen piirilevylle suunniteltiin 4-napainen liitin hinauskoukkua varten. Liittimenä käytettiin saman sarjan 4-napaista versiota, kuin servoille käytetty 3-napainen liitin. Käyttöjännitteen ja maan lisäksi siinä on pinnit koukun asennon mittausta varten ja hinaussiiman esteen tilan tai koukkuun kohdistuneen hinaussiiman voiman mittausta varten. Nämä on johdettu mikroprosessorin i/o-pinneille, jälkimmäinen pinnille, jota voidaan käyttää myös analogisena sisääntulona.

Jos hinauskoukku on mekaaninen, ilman servotoimista hinaussiiman estettä, i/o-pinnejä käytetään tavanomaisina digitaalituloina mikroprosessorin sisäisiä ylösvetovastuksia hyväksikäyttäen, jolloin hinauskoukun mikrokytkimen toimiessa i/o-pinni maadoittuu. Jos hinauskoukussa on voiman mittaus, kyseistä i/o-pinniä käytetään analogiatulona, eikä sisäistä ylösvetovastusta kytketä päälle. Kytkentävärähtelyjen vaimentamiseksi hinauskoukun tilatietojen mittaustuloihin suunniteltiin yksinkertaiset RC-suodattimet.

Hinaussiiman voiman mittaus toteutettiin venymäliuska-antureiden avulla, jotka mittaavat hinauskoukun vääntymää. Vääntymää mitataan kahdella venymäliuskalla, joista toinen on taipuvan osan toisella puolella ja toinen toisella puolella.

Venymäliuska-anturi on mittauselementti, jonka resistanssi muuttuu, kun sen muoto muuttuu. Sen toiminta perustuu anturissa olevan metallijohteen venymisen seurauksena muuttuvaan resistanssin arvoon. [9]

Venymäliuskat muodostavat kahden vastuksen kanssa Wheatsonen siltakytkennän, niin sanotun puolisillan, jonka yli olevaa jännitettä mitataan. Wheatstonen siltakytkentää käytetään tuntemattoman vastuksen selvittämiseksi. Se muodostuu jännitelähteestä ja neljästä vastuksesta, jotka kytketään neliön muotoon. Jännite syötetään siltakytkennän kahteen vastakkaiseen kulmapisteeseen ja kahden muun kulman välillä olevaa jännitettä mitataan. Jos vastukset ovat samansuuruisia, mitattu jännite on nolla, mutta jos ne ovat erisuuruiset, mitattu jännite poikkeaa nollasta. Kun siltakytkennässä on kaksi venymäliuskaa, saadaan kompensoitua lämpölaajenemisen vaikutus mittaustulokseen, sillä lämpölaajenemisen seurauksena molempien venymäliuskojen resistanssi muuttuu saman verran, eikä mitattava jännite muutu. [9] Kuvasta 11 nähdään venymäliuskojen puolisiltakytkennän periaatekuva.

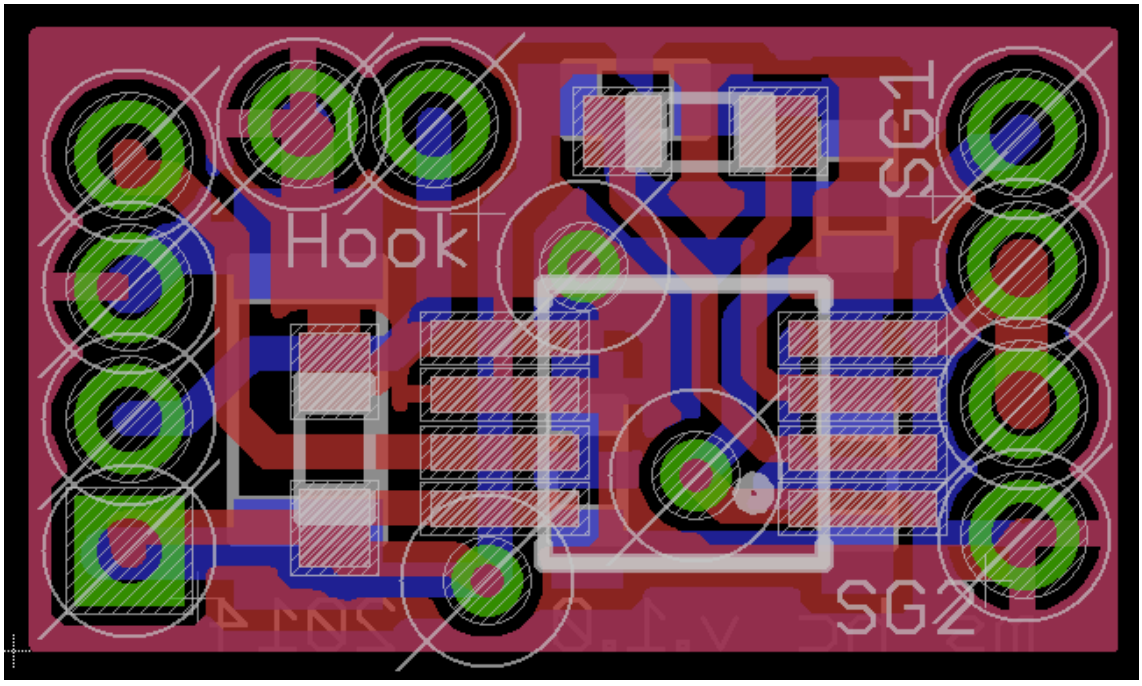


Kuva 11 Venymäliuskat puolisiltakytkennässä

Kun hinauskoukun joustava kohta taipuu, toisella puolella oleva venymäliuska venyy ja sen resistanssi kasvaa samaan aikaan kun toinen venymäliuska painuu kasaan ja sen resistanssi pienenee. Venymäliuskojen vastuksien muuttuessa siltakytkennän mitattava jännite poikkeaa nollasta. Käytettäessä kahta venymäliuskaa yhden sijaan, myös mittausherkkyys on kaksinkertainen verrattuna yhden liuskan kytkentään.

Mitatun jännitteen muuttamiseksi mikroprosessorin analogiatulon mittausalueelle sopivaksi käytetään instrumentointivahvistinta. Instrumentointivahvistimella on korkea tuloimpedanssi ja se vaimentaa yhteismuotoisia häiriöitä tehokkaasti, joten se sopii hyvin tarkkuutta vaativiin mittauksiin. Instrumentointivahvistimen vahvistuta voidaan muuttaa usein yhden vastuksen avulla.[9]

Instrumentointivahvistimelle suunniteltiin oma pieni piirilevy, joka voidaan sijoittaa hinauskoukkuun mahdollisimman lähelle venymäliuskoja häiriöiden minimoimiseksi. Siltakytkennän kaksi vastusta mitoitettiin vastaamaan käytettyjen venymäliuskojen lepotilan resistanssia, jolloin hinauskoukun ollessa rasittamaton, mittaussillan yli oleva mitattu jännite pitäisi olla nolla. Instrumentointivahvistimeksi valittiin Analog devices:n AD8226, jonka vahvistusta voidaan säätää yhdellä ulkoisella vastuksella, ja jonka tyypillinen virrankulutus on noin 350µA. Piirilevylle suunniteltiin myös kytkentäpiste hinauskoukun asentoa mittaavalle mikrokytkimelle, joten liitäntä timerin piirilevylle onnistuu yhdellä 4-napaisella kaapelilla. Kuvasta 12 nähdään instrumentointi-vahvistimelle suunniteltu piirilevy Eagle:n näkymässä. Instrumentointivahvistimen piirilevyn kytkentäkaavio on esitetty liitteessä B.



Kuva 12 Instrumentointivahvistimen piirilevy Eagle:n näkymässä

3.6 Bluetooth-moduuli

Jotta lennokinohjainta voitaisiin säätää tietokoneella, täytyy tietokoneen ja ohjaimen pystyä kommunikoimaan. Yhteys toteutettiin sarjaliikenneyhteytenä bluetoothin välityksellä. Bluetooth-yhteys toteutettiin käyttämällä valmista HC-06 bluetooth-moduulia.

HC-06 on bluetooth 2.0 versiota tukeva bluetooth-sarjaliikennemuunninmoduuli. Sen käyttöjännite on 3,3V ja se on pysyvästi asetettu slave-tilaan. Se tukee sarjaliikenteen nopeuksia aina 1382400 bit/s asti, mutta suositus on käyttää korkeintaan 115200 bit/s nopeutta. [10]

Käyttöjännitteiden lisäksi bluetooth-moduuli tarvitsee yhteyden mikroprosessorin sarjaliikennenapoihin. Bluetooth-moduulissa on myös esiohjelmoitu i/o-lähtö ledille, joka ilmaisee missä tilassa moduuli on. Tähän lähtöön kytkettiin led esivastuksen kautta bluetooth-moduulin maahan. Led-lähtö on ohjelmoitu toimimaan siten, että moduulin etsiessä paria, led vilkkuu ja kun pari on muodostettu, led palaa jatkuvasti.

Bluetooth-moduulia tarvitaan ainoastaan timerin ollessa ohjelmointitilassa, joten muulloin se on ainoastaan turhana kuormana. Koska HC-06 bluetooth-moduulia ei ole mahdollista ohjata ohjelmallisesti pois päältä, toteutettiin sen päälle- ja poiskytkentä johtamalla sen maaliittimet transistorikytkimen kautta maahan. Transistorina käytettiin mosfet-tyyppistä transistoria ja sen hilalle kytkettiin alasvetovastus.

3.7 Radiomoduuli

HopeRF:n valmistama RFM22B on integroitu ISM-taajuuskaistalla toimiva radiolähetinmoduuli. ISM-taajuusalue on maailmanlaajuinen radiotaajuuskaista, joka on alun perin luotu teolliseen, tieteelliseen ja lääketieteelliseen käyttöön (Industrial, Scientific, Medical). ISM-taajuusalueita on yhteensä 12 alkaen 6,765 MHz:sta päättyen 246 GHz:iin.[11] ISM-taajuuskaistalla toimivat pienitehoiset lähettimet eivät tarvitse erillistä lupaa. Taajuushyppelyllä tarkoitetaan tekniikkaa, jossa lähettimen kantoaallon taajuutta vaihdellaan nopealla tahdilla vastaanottimen tuntemalla tavalla.[15] RFM22B-moduulin tukemat taajuuskaistat ovat 433, 470, 868 ja 915 MHz ja se tukee ns. taajuushyppelyä. Moduuli voidaan ohjata lähetys- ja vastaanottotilojen lisäksi useisiin eri toimintatiloihin, joilla voidaan optimoida virrankulutusta. Käyttöjännite voi olla 1.8 – 3.6 V ja lähetystehoa voidaan ohjelmallisesti säätää välillä 1-20 dBm, vastaanottimen herkkyyden ollessa -121 dBm. Kommunikointi mikrokontrollerin kanssa tapahtuu SPI-väylän välityksellä. [11]

SPI-väylä on synkronoitu sarjaväylä kahden tai useamman laitteen välille. SPI-väylässä on kellosignaali, luku- ja kirjoitussignaalit sekä 0-aktiivinen slave-laitteen valinta-signaali. Väylällä voi kommunikoida vain yksi master-laite, mutta se voi välittää tietoa

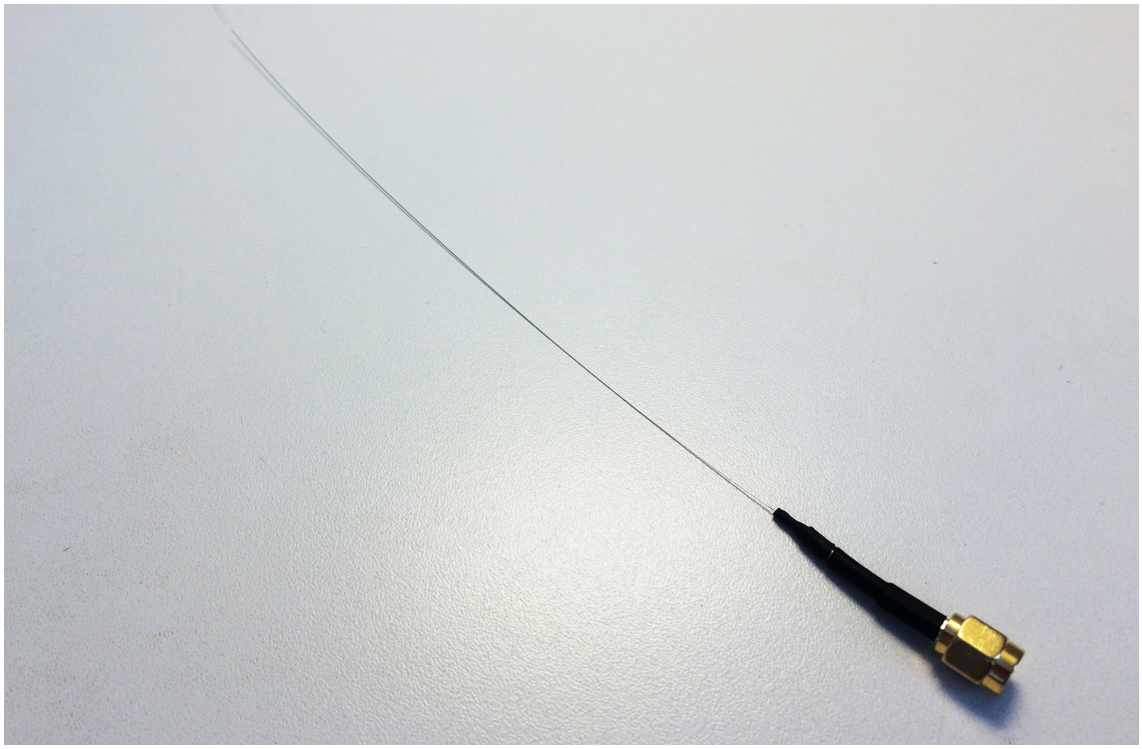
useammalle slave-laitteella samalla kertaa. SPI-väylän kommunikointi muodostuu 16 bitin mittaisista jaksoista, joka koostuu luku/kirjoitus bitistä, 7-bittisestä osoitteesta ja 8:sta data-bitistä. [12]

SPI-väyläsignaalien lisäksi radiomoduuli kytkeytyy mikroprosessoriin nIRQ-signaalilla, mikä on radiomoduulin keskeytystapahtuman ilmaiseva ulostulo. Jos keskeytys on tapahtunut nIRQ-pinni maadoittuu ja SPI-väylän välityksellä voidaan käydä lukemassa radiomoduulin keskeytysrekisteri. OpenLRS-ohjelmisto käyttää radiomoduulin keskeytyksiä toimintoihinsa.

OpenLRS-laitteista käytetään RFM22b-radiomoduulia ja pienen virrankulutuksen ja pienen fyysisen koon ansiosta se soveltuu hyvin käytettäväksi myös ms-timerissa.

3.8 Lennokin ohjaimen radioantenni

Kaikki radiolähettimet ja vastaanottimet tarvitsevat antennin. Antennilla pyritään siirtämään lähetysteho mahdollisimman tehokkaasti vapaaseen tilaan ja päinvastoin. Jotta antenni toimisi tehokkaasti olisi sen hyvä olla puoliaallon pituinen tai sen moninkerta. Radio-aallon pituus saadaan jakamalla aallon nopeus sen taajuudella. Antennin suorituskykyyn vaikuttavaa myös moni muu asia, kuten sen säteilykuvio ja vahvistus sekä antennin piiriominaisuudet kuten impedanssi, hyötysuhde ja kaistanleveys. [13]



Kuva 13 Lennokin ohjaimen antenni

Lennokin ohjaimen antennin vaatimuksina ovat tehokkuuden lisäksi kompakti koko, kevyt paino ja pieni ilmanvastus, sillä se täytyy tuoda lennokin hiilikuituisen rungon ulkopuolelle. Monopoli- eli maatasoantenni vastaa hyvin ainakin jälkimmäisiin ehtoihin, ja sen soveltuvuutta lennokinohjaimen antenniksi päätettiin testata.

Maatasoantenni eli monopoliantenni on ympärisäteilevä antennityyppi, jossa jokin johtava taso tai pinta toimii antennin toisena elementtinä ja toisena elementtinä tästä tasosta eristetty putki tai metallilanka. [13]

Tehokkaasti toimiakseen antennin täytyy olla puoliaallon tai sen moninkerran pituinen. Tyhjiössä radioaallot liikkuvat valonnopeudella (300000 km/s) ja jakamalla se käytetyn radio-moduulin taajuudella (435 Mhz) saadaan aallon pituudeksi n. 689,7 mm. Puoliaaltoantenni olisi liian pitkä, joten päätettiin kokeilla 1/4-aaltoantennin toimintaa. Neljännesaallon pituus on n. 172,35 mm, joten antennista tehtiin sen pituinen. Kuvassa 13 on käytetty antenni, se liitetään timerin piirilevylle SMA-tyyppisellä liittimellä.

4. OHJELMISTO

Jotta lennokin ohjain pystyisi suorittamaan muun muassa halutut i/o-porttien toiminnot, sarjaliikenteen ja muistinhallinnan, täytyy siihen olla tallennettuna konekielinen ohjelma. Jotta kommunikointi tietokoneen kanssa onnistuisi, täytyy myös tietokoneella olla ohjelma, jolla voidaan muuttaa säädettäviä parametreja ja lähettää ne tietyn kaavan mukaisesti sarjaliikenteen välityksellä lennokin ohjaimelle sekä lukea tallennetut parametrit ohjaimelta.

4.1 Lennokin ohjaimen ohjelma

Lennokin ohjaimen ohjelmassa sen toimintatilat on jaettu eri osiin, riippuen siitä, missä tilanteessa lennon aikana, ennen tai jälkeen lennon ollaan. Lennokin ohjelmisto on kehitetty Arduino-kehitysympäristössä, joka on avoimen lähdekoodin mikrokontrolleri-kehitysalusta ja ohjelmointiympäristö.

4.1.1 Arduino

Arduino on alun perin kehitetty opiskelijoita varten Olivett:lla vuonna 2005. Aluksi Arduino-laitteita valmisti italialainen Smart Projects, mutta koska tekninen määrittely on julkinen ja vapaasti käytettävissä, myöhemmin niitä on valmistaneet muutkin tahot. Arduino perustuu 8-bittisiin Atmel AVR-mikrokontrollereihin ja ohjelmointi tapahtuu omassa kehitysympäristössä Arduino IDE:ssä (Integrated Development Envinroment). Arduino-kehitysympäristöön on saatavana lukuisia valmiita ohjelmakirjastoja. Arduino-laitteiden mikroprosessorille on valmiiksi ladattu Arduinon oma niin sanottu bootloader, jonka avulla oma ohjelma voidaan kirjoittaa mikroprosessorin flash-muistiin käyttäen sarjaliikennettä. [14]

Arduino-ohjelmat kirjoitetaan C- tai C++ -ohjelmointikielellä. Arduino IDE sisältää vakiona Wiring-nimisen ohjelmakirjaston, joka sisältää muun muassa yksinkertaistetut käskyt i/o-operaatioille. Arduino ohjelmassa on kaksi pääfunktiota, funktiot *setup* ja *loop*. Funktiota *setup* suoritetaan kerran laitteen käynnistymisen jälkeen ja sen jälkeen *loop*-funktiota toistetaan siihen asti, kunnes laite sammutetaan. [14]

Valmiita Arduino-laitteita on joukko erilaisia, jotka eroavat toisistaan mm. muistien määrissä, digitaali- ja analogiapinnien määrissä sekä sarjaliikenteen ja muiden yhteyksien toteutustavoissa. Eri käyttötarkoitukseen on erimuotoisia ja -kokoisia laitteita, kuten mm. kankaaseen ommeltavia malleja. Lisäksi Arduino-laitteisiin on

saatavana laajennusosia eli ”shieldejä”, joilla peruslaitteeseen voi helposti lisätä ominaisuuksia, esimerkiksi GSM- tai WiFi-yhteyden. [14]

Lennokin ohjaimessa ei hyödynnetty valmiita Arduino-laitteita, mutta sen ohjelmisto kehitettiin käyttäen Arduinon kehitysympäristöä ja joitain valmiita ohjelmistokirjastoja sekä vapaan lähdekoodin OpenLRS-ohjelmistoa.

4.1.2 OpenLRS

OpenLRS on vapaan lähdekoodin pitkän kantomatkan järjestelmä (Long Range System), joka on tarkoitettu mm. radio-ohjattavien laitteiden ja robottien harrastajille. OpenLRS laitteet sisältävät RFM22B-radiomoduulin ja Atmega328-mikroprosessorin, johon on ladattu Arduino bootloader. [15]

Alkuperäinen OpenLRS-ohjelma tukee kolmen kanavan taajuushyppelyä 50 Hz:n virkistystaajuudella. OpenLRS-laitteet voivat toimia sekä lähettimenä, että vastaanottimena, mikä mahdollistaa kaksisuuntaisen radioliikenteen eli niin sanotun telemetrian, jonka avulla voidaan lähettää lennokista tietoja radiolähettimelle, esimerkiksi akkujännite tai lentokorkeus. Vastaanottimissa on I2C-väyläliitin, johon voidaan kytkeä lisälaitteita kuten vaikkapa korkeusmittari tai kiihtyvyysanturi. [15]

Ms-timer perustuu osittain OpenLRS:ään sekä laitteiston että ohjelmiston osalta. Ohjelmistossa sovelletaan radioliikenteessä alkuperäistä OpenLRS-koodia mikroprosessorin ja radiomoduulin väliseen kommunikointiin ja servo-ohjauksiin.

Alkuperäisessä OpenLRS-koodissa servokanavien luku tapahtuu jatkumona ohjelman pääsilmukassa. Ms-timerin ohjelmassa radio-moduulin hallinta on siirretty omaan funktioonsa, jota kutsutaan tarvittaessa. Alkuperäisen koodissa on ns. failsafe-toiminta, joka radioyhteyden katketessa asettaa servot ennalta säädettyihin failsafe-asentoihin. Alkuperäistä failsafe-koodia on muokattu siten, että mikäli yhteyttä lähettimeen ei ole, asetetaan servokanavien arvot keskiasentoon, eikä ne tällöin vaikuta timerin toimintaan.

Servojen ohjauspulssit muodostetaan OpenLRS-koodissa mikroprosessorin Timer/Counter1:n keskeytysrutiinissa. Timer/Counter1 on asetettu suorittamaan keskeytyksen sen laskiessa 20000:een. Laskurin kasvatus on asetettu tapahtuvaksi 1/8 kellotaajuudella, jolloin keskeytys tapahtuu 20 ms:n välein. Timer/Counter1:n keskeytysrutiinin ohjelmakoodi on esitetty algoritmissa 1, siinä kaikki servolähdöt asetetaan ensin nollaksi ja jokainen servolähtö nostetaan vuorollaan ylös ja asetetaan laskurin arvoksi 20000 ja vähennetään siitä ohjauspulssin leveys mikrosekunteina. Uuden keskeytyksen tullessa lähdöt asetetaan taas nollaan. Kun kaikki servot käyty läpi, asetetaan laskurin arvoksi 20000 ja vähennetään siitä servopulssit yhteensä, jolloin pulssituksille saadaan 50 Hz:n taajuus.

```
ISR(TIMER1_OVF_vect)
{
  static unsigned int ms; //milliseconds
  unsigned int us; // this value is 1 microseconds
  Servo_Ports_LOW;
  Servo_number++; // jump to next servo

  if (Servo_number>3) // back to the first servo
  { // Led_beacon_on will be true for 60 ms in every 3 seconds
    ms += 20;
    if (ms >= 3000 && !Led_beacon_on)
    {
      Led_beacon_on = true;
      ms = 0;
    }
    if (Led_beacon_on && ms >= 60)
    {
      Led_beacon_on = false;
    }
    Total_ppm_time = 0; // clear the total servo ppm time
    Servo_number=0;
  }
  if (Servo_number == 3)
  { // Servos accepting 50hz ppm signal, this is why we are
    // waiting for 20ms before second signal brust.
    us = 20000 - Total_ppm_time;
  }
  else
  us = Servo_position[Servo_number]; // read the servo timing from buffer
  Total_ppm_time += us; // calculate total servo signal times.

  switch (Servo_number)
  {
    case 0:
      Servo1_OUT_HIGH;
      break;
    case 1:
      Servo2_OUT_HIGH;
      break;
    case 2:
      Servo3_OUT_HIGH;
      break;
    case 3:
      Servo4_OUT_HIGH;
      break;
  }
  TCNT1 = 20000 - us; // configure the timer interrupt for X micro seconds
}
```

Algoritmi 1 Timer/Counter1:n keskeytysrutiini

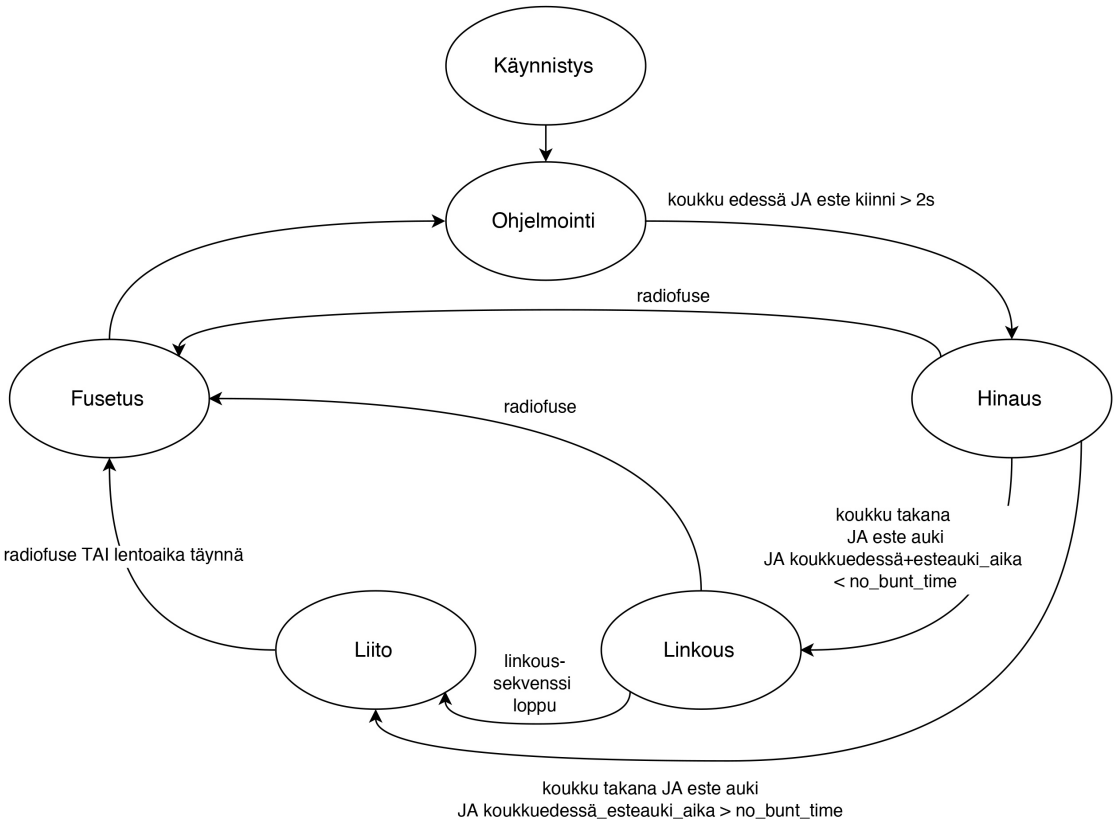
Keskeytysrutiiniin on ms-timerissa lisätty millisekunteja laskeva muuttuja, jonka perusteella päivitetään muuttujan *Led_beacon_on* arvoksi tosi 60 ms:n ajaksi aina kolmen sekunnin välein. Tätä muuttujaa käytetään led-majakan vilkutukseen ja suorittamalla sen tilan päivitys kellokeskeytyksessä saadaan led-majakan vilkutus tehtyä ilman, että se vaikuttaa timerin muihin ajoituksiin.

4.1.3 Lennokin ohjaimen ohjelman rakenne

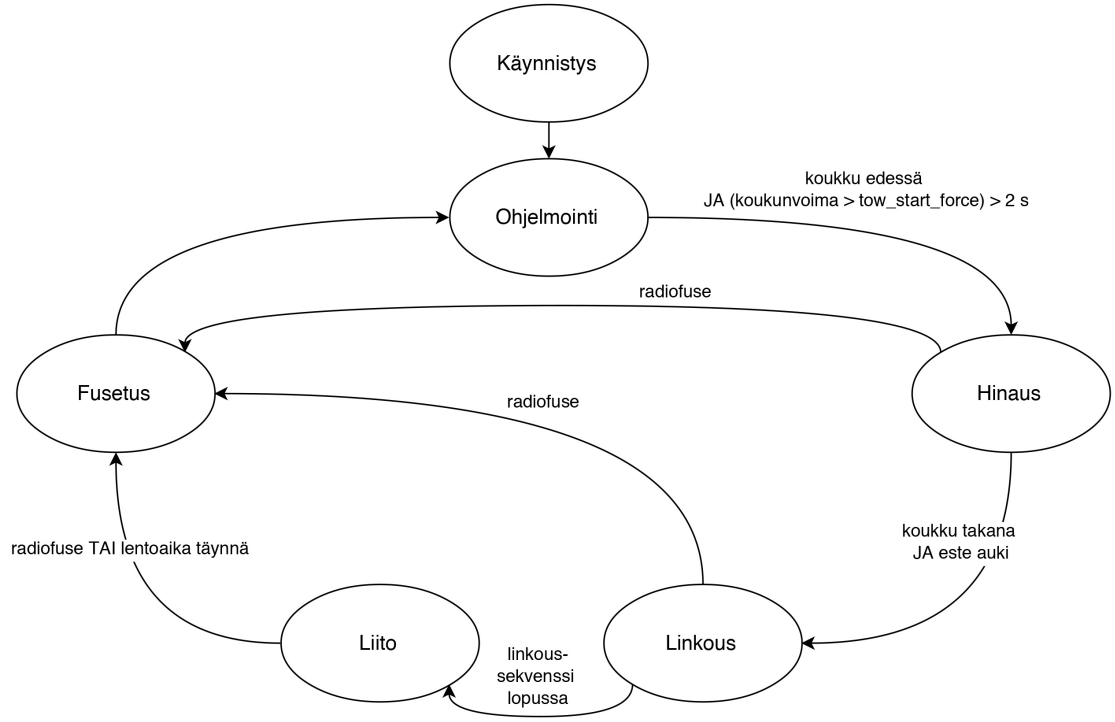
Lennokin ohjaimen ohjelma on jaettu useaan eri tiedostoon ohjelman selkeyden ja luettavuuden parantamiseksi. Tiedosto *ms_timer* sisältää funktioiden *setup* ja *loop* lisäksi kellokeskeytysrutiinin sekä sarjaliikennefunktion *serial_event*, jota kutsutaan ohjelmointitilassa jos sarjaliikennepuskurissa on käsiteltävää dataa. Globaalien muuttujien määrittelyt ja define:t on sijoitettu *config.h*-tiedostoon. Tiedosto *functions* sisältää mm. muistinhallintaan liittyviä yleiskäyttöisiä funktiota ja laskentafunktioita, joilla tietokoneelta saadut servojen asentotiedot muutetaan ohjauspulssinpituuksiksi ja päinvastoin, sekä timerin toiminnallisuuteen liittyviä funktioita, kuten esimerkiksi linkousfunktio *launch*. Tiedostossa *pointers* määritellään muistiosoitteet osoitinmuuttujille, joiden avulla luetaan ja kirjoitetaan mikroprosessorin EEPROM-muistiin. Tiedosto *rfm22b* sisältää funktion *read_rx* sekä alkuperäiset OpenLRS-ohjelman funktiot. Funktio *read_serial* on funktio, jossa suoritetaan sarjaliikenteen toimintoja ja se on sijoitettu tiedostoon *serial.*

Lennokin ohjaimen ohjelma toteutettiin siten, että käynnistymisen jälkeen se on aina jossain tilassa, joka on määritelty muuttujassa *Timer_mode*. Ohjelman eri tiloja ovat ohjelmointi *(Programming),* hinaus *(Tow),* linkous *(Launch),* liito *(Glide)* ja fusetus *(Dt)*. Ohjelmointi- ja hinaustilojen toiminnat jakautuvat kahteen eri osaan, riippuen onko käytössä servotoiminen hinauskoukku vai perinteinen mekaaninen koukku. Koukun tyyppi määritellään lennokin ohjaimelle parametrina tietokoneohjelmasta käsin.

Pääsilmukassa koodi on jaettu if-else rakenteella haarautumaan sen mukaan, mikä ohjelman tila on päällä. Muuttujaa *Timer_mode* muutetaan tiettyjen ehtojen mukaisesti siten, että ohjelmointitilasta siirrytään aina hinaustilaan ja hinaustilasta linkoustilaan tai liitotilaan. Liitotilasta siirrytään fusetustilaan ja fusetustilasta takaisin ohjelmointitilaan. Lisäksi fusetustilaan voidaan siirtyä suoraan mistä tahansa tilasta, jos on tehty radiofusetus. Tilasiirtymät eroavat hieman, riippuen käytössä olevasta hinauskoukun tyypistä. Kuvassa 14 on esitetty tilakaavio mekaanisella hinauskoukulla ja kuvassa 15 servotoimisella hinauskoukulla, jossa on hinaussiiman voiman mittaus.



Kuva 14 Lennokin ohjaimen tilakaavio mekaanisella hinauskoukulla



Kuva 15 Lennokin ohjaimen tilakaavio servokoukulla

4.1.4 Käynnistys

Timerin käynnistyessä määritetään i/o-porttien suunnat ja asetetaan tarvittaessa input-porttien sisäiset ylösvetovastukset. Tämän jälkeen määritetään muistiosoitteet osoitinmuuttujille ja luetaan niiden perusteella EEPROM:iin tallennetut säätöparametrit omiin muuttujiinsa ja taulukoihinsa.

Sarjaliikennettä varten alustetaan mikrokontrollerin sarjaliikenneportti, sekä SPI-väylä radiomoduulin toimintoja varten. Seuraavaksi alustetaan radiomoduuli ja Timer/Counter1:n kellokeskeytys OpenLRS-funktioita käyttäen ja lopuksi sallitaan keskeytykset globaalisti.

Servojen asentotiedot määritellään tietokoneohjelmassa välillä -125…125 ja lähetetään timerille positiivisena kokonaislukuna 0…250. Asentotiedot muutetaan servojen ohjauspulssien pituuksiksi ja skaalataan tietokoneohjelmassa asetettujen liikeratojen rajojen mukaisesti. Algoritmissa 2 on esitetty funktio *convert_to_us*, jolla tieto-koneohjelmasta saatu asentoparametri muutetaan mikrosekunneiksi.

```
 1 unsigned int convert_to_us(int s_pointer, int s_cntr, int s_rev,
 2                            int s_lim_pos, int s_lim_neg)
 3 { // from laptop, servo positions we get are between 0...250
 4   long temp = (read_8bit_eeprom(s_pointer)-125) * 5;
 5   if (temp >= 0)
 6   {
 7     temp = temp * s_lim_pos / 125;
 8   }
 9   else
10   {
11     temp = temp * s_lim_neg / 125;
12   }
13   return (temp + s_cntr) * s_rev + 1500;
14 }
```

Algoritmi 2 Funktio convert_to_us

Funktiolle *convert_to_us* annetaan parametrina servon asennon sisältämän muistinpaikan osoite, servon keskikohdan säätöarvo, servon liikkeen peilausasetus ja liikealueen rajat. Keskikohdan säätöarvo on yksinkertainen offset-säätö, joka vaikuttaa koko servon liikerataan riippumatta liikeradan rajoista. Algoritmissa 2 nähdään rivillä 4 kuinka 0…250 välillä oleva servon asetus muutetaan ensin mikrosekunneiksi välillä -625…625, jonka jälkeen saadut arvot skaalataan etumerkistä riippuen joko positiivisella tai negatiivisella liikealueen raja-arvolla. Lopuksi saatuun arvoon lisätään muuttujan *s_cntr* sisältämä offset, kerrotaan muuttujalla *s_rev* ja lisätään 1500, jolloin tuloksena on pulssin pituus välillä 725…2275 µs. Liikealueen raja-arvoille voidaan asettaa arvoja välillä 0…150, skaalauskertoimen jakajan ollessa 125, joten halutessa saadaan aikaan hieman normaaleja 1 - 2 ms:n pulsseja pidempiä tai lyhyempiä

ohjauspulsseja. Joissain tilanteissa yli- tai alimittaisista ohjauspulsseista voi olla hyötyä riippuen servotyypistä, mutta kaikkien servojen liikeradat eivät välttämättä niille riitä.

4.1.5 Ohjelmointitila

Ohjelmointitilassa tarkkaillaan koko ajan sarjaliikennepuskuria, ja jos sinne tulee käsiteltävää dataa, suoritetaan funktio *serial_event.* Sarjaliikenteestä luetaan ensin tietokoneen lähettämä käsky muuttujaan *Serial_command* ja sen jälkeen mahdollisesti tuleva data taulukkoon odottamaan käsittelyä funktiossa *read_serial*.

Radio-ohjauksella tehtävät säädöt tallennetaan aina ensin ohjaimen SRAM-muistiin. Tietokoneohjelmalla voidaan lukea joko ohjaimen SRAM-muisti tai EEPROM-muisti, jolloin voidaan verrata alkuperäisiä ja muutettuja säätöjä. Kun muutetut säädöt on hyväksytty, voidaan ne kirjoittaa ohjaimen EEPROM-muistiin.

Ohjelmointitilassa suoritettavassa funktiossa *read_serial* ohjelman toiminta haarautuu muuttujan *Serial_command* sisältämän sarjaliikennekäskyn mukaisesti. Muuttujan saadessa arvot *Read_tow*, *Read_launch, Read_glide* tai *Read_setup* luetaan sarjaliikenteestä muuttujan arvon mukaisen tietokoneohjelman välilehden sisältämät asetukset timerin EEPROM-muistiin. Käskyllä *Read_rx_setup* timer lukee radiomoduulin asetukset ja *Pincode*- ja *Devicename* -käskyillä se asettaa bluetooth-moduulin pin-koodin tai nimen halutuksi. Käsky *Test_launch* on linkousliikesarjan testausta varten ja *Test_on*- ja *Test_off* -käskyillä ohjataan servojen liikeratojen testaustila päälle tai pois päältä.

Lennokin ohjaimesta voidaan lukea asetukset EEPROM-muistista tai SRAM-muistin sisältämien muuttujien muutokset, jos niitä on säädetty radio-ohjauksella. Nämä tapahtuvat sarjaliikennekäskyillä *Read_eeprom* ja *Read_ram*. Akkujännite puolestaan luetaan tietokoneohjelmaan *Battery*-käskyllä ja *Establish*-käskyllä ei tehdä muuta kuin lähetetään kuittaus takaisinpäin, jotta tietokoneohjelma tietää sarjaliikenneyhteyden olevan kunnossa. Sarjaliikennekäskyn ollessa *None* ei tehdä mitään.

Riippuen hinauskoukun tyypin määrittelystä tarkkaillaan ohjelmointitilassa koukun asentotietoja ja mahdollisesti koukkuun kohdistuvaa voimaa. Mekaanisella hinauskoukulla ohjelmointitilasta siirrytään hinaustilaan, kun hinauskoukun este on kiinni ja koukku on ollut yhtäjaksoisesti etuasennossa yli kahden sekunnin ajan. Vastaavasti jos käytössä on servokoukku, koukun ollessa etuasennossa ja siihen kohdistuvan voiman ylittäessä asetetun arvon yli kahden sekunnin ajan, ohjataan hinaussiiman este kiinni ja siirrytään hinaustilaan. Ajanmittaus tehdään Arduinon funktiolla *millis*, joka laskee millisekunteja ohjelman käynnistyksestä lähtien. Tallentamalla ajanmittauksen aloitushetken millisekuntilukema ja vertaamalla sitä myöhemmän hetken millisekuntilukemaan, saadaan erotuksena laskettua kulunut aika.

Funktio *millis*:n palauttama arvo on 32-bittinen kokonaisluku ja se ylivuotaa noin 50 vuorokauden välein.

4.1.6 Hinaustila

Radio-ohjaus ei ohjaa ohjainpintoja absoluuttisesti vaan ohjauksella on tarkoitus hienosäätää ennalta asetettuja ohjainpintojen asentoja. Radio-ohjauksen avulla ainoastaan mitataan radio-lähettimen ohjaussauvojen poikkeama keskiasennosta ja lisätään poikkeama ennalta säädettyyn servonasentoon. Koska lennokkiradiolähetin on osittain mekaaninen laite, ei ohjainsauvojen asentotieto ole aina välttämättä täsmälleen sama ohjaussauvojen ollessa keskitettynä, vaan pieniä poikkeamia esiintyy. Tästä syystä siirryttäessä ohjelmointitilasta hinaustilaan, luetaan radio-ohjauksen ohjaussauvojen asentotiedot lennokin ohjaimen muistiin referenssiarvoiksi, joihin vertaamalla hetkittäisiä arvoja saadaan laskettua ohjaussauvan poikkeama keskipisteestä. Tästä syystä radio-lähettimen ohjaussauvat tulee olla keskitettyinä hinaustilaan siirryttäessä.

Hinaustilan toiminta jakautuu kahteen eri osaan sen mukaan, onko käytössä mekaaninen hinauskoukku vai voiman mittauksella varustettu servokoukku. Mikäli käytössä on voiman mittauksella varustettu koukku, lennokin ohjainpintoja (sivuperäsin, korkeus-peräsin ja sisäsiiven asetuskulma eli wiggler) ohjataan koukun tilatiedon ja hinaussiiman vetovoiman perusteella. Mekaanisella hinauskoukulla ohjainpintojen asennot määräytyvät ainoastaan hinauskoukun tilatiedon perusteella. Hinauskoukun tiloja on kolme, koukku edessä ja este kiinni, koukku takana ja este kiinni sekä koukku edessä ja este auki.

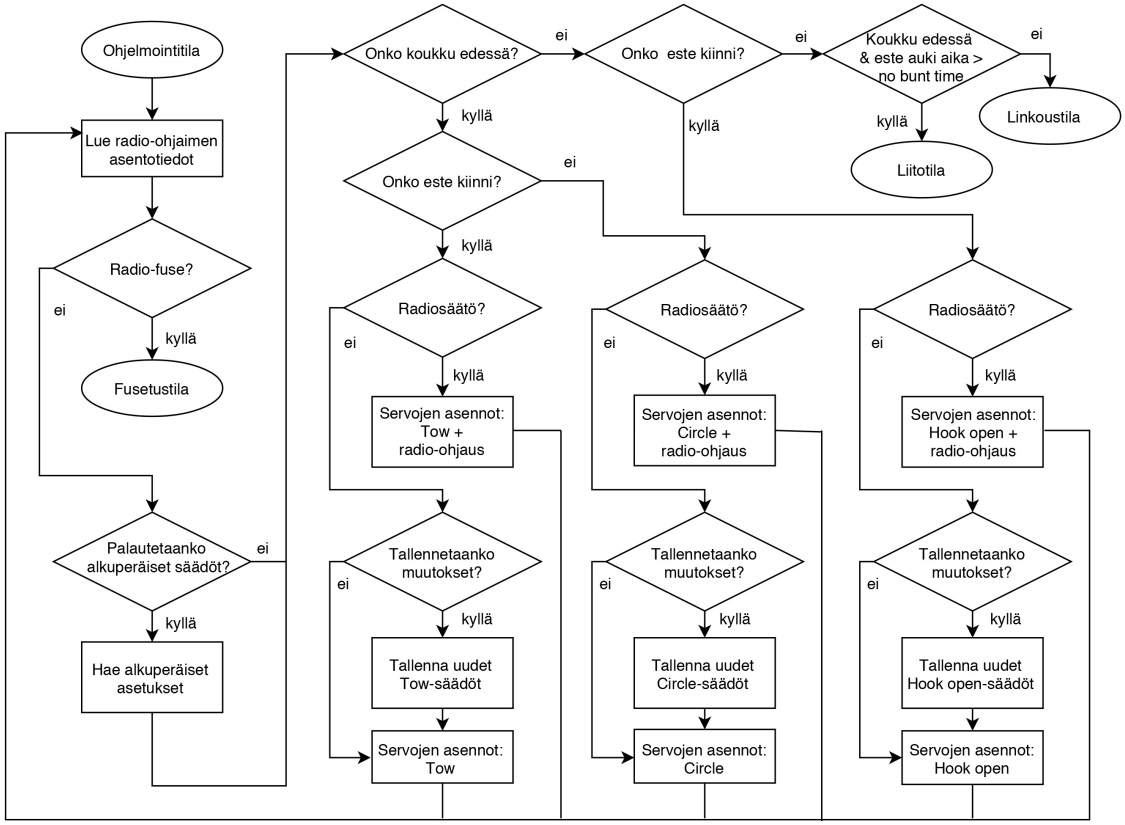
Mekaanisella koukulla ohjainpintojen asetuksia on vain yhdet jokaista hinauskoukun tilaa kohti, mutta jos käytössä on servokoukku, jokaiselle tilalle on määritettävissä viisi eri ohjainpintojen asentoa, joiden välillä liikutaan sen mukaan, kuinka suuri on hinaussiiman voima. Ohjainpintoja liikuttavien servojen asennot ja hinaussiiman voimien asetusarvot ovat tietokoneohjelman avulla määritettäviä parametreja.

Kuvassa 16 on esitetty hinaustilan lohkokaavio, kun käytössä on mekaaninen hinauskoukku. Jokaisella kierroksella tarkastetaan, onko tehty radiofusetus ja jos on, muutetaan timerin tila fusetustilaksi. Mikäli radiofusetusta ei ole tehty, tarkastetaan onko pyydetty palauttamaan alkuperäiset säädöt EEPROM-muistista. Tämän jälkeen ohjelma haarautuu kolmeen haaraan koukun tilan mukaan. Jokaisessa haarassa tarkastetaan, onko radiosäätö päällä, ja mikäli on, kyseisen tilan servojen asentoihin lisätään radio-ohjauksen vaikutus.

Kun radiosäätö kytketään pois päältä, tallennetaan sen hetkiset servojen asennot kyseisen hinaustilan servojen asennot sisältäviin muuttujiin, mikäli radiosäätö on kytketty päälle ja pois saman hinaustilan ollessa aktiivinen. Jos radiosäätö kytketään

pois eri hinaustilan ollessa aktiivinen, ei säätöjä tallenneta, jottei vahingossa tehdä hallitsemattomia säätömuutoksia.

Hinaustilasta siirrytään joko linkoustilaan tai liitotilaan, kun hinauskoukun este on auennut ja koukku liikkuu taka-asentoon. Valinta liitotilan ja linkoustilan välillä tehdään sen perusteella, kuinka kauan vaihe, jossa koukku on edessä ja este auki, on kestänyt. Jos tämän vaiheen kesto on yli asetetun ajan, siirrytään suoraan liitotilaan, koska usein tämän kaltainen tilanne johtuu hinauskoukun esteen tahattomasta avautumisesta. Tällöin turvallisempaa on siirtyä suoraan liitotilaan, kuin ajaa linkoustilan liikesarjat servoille.



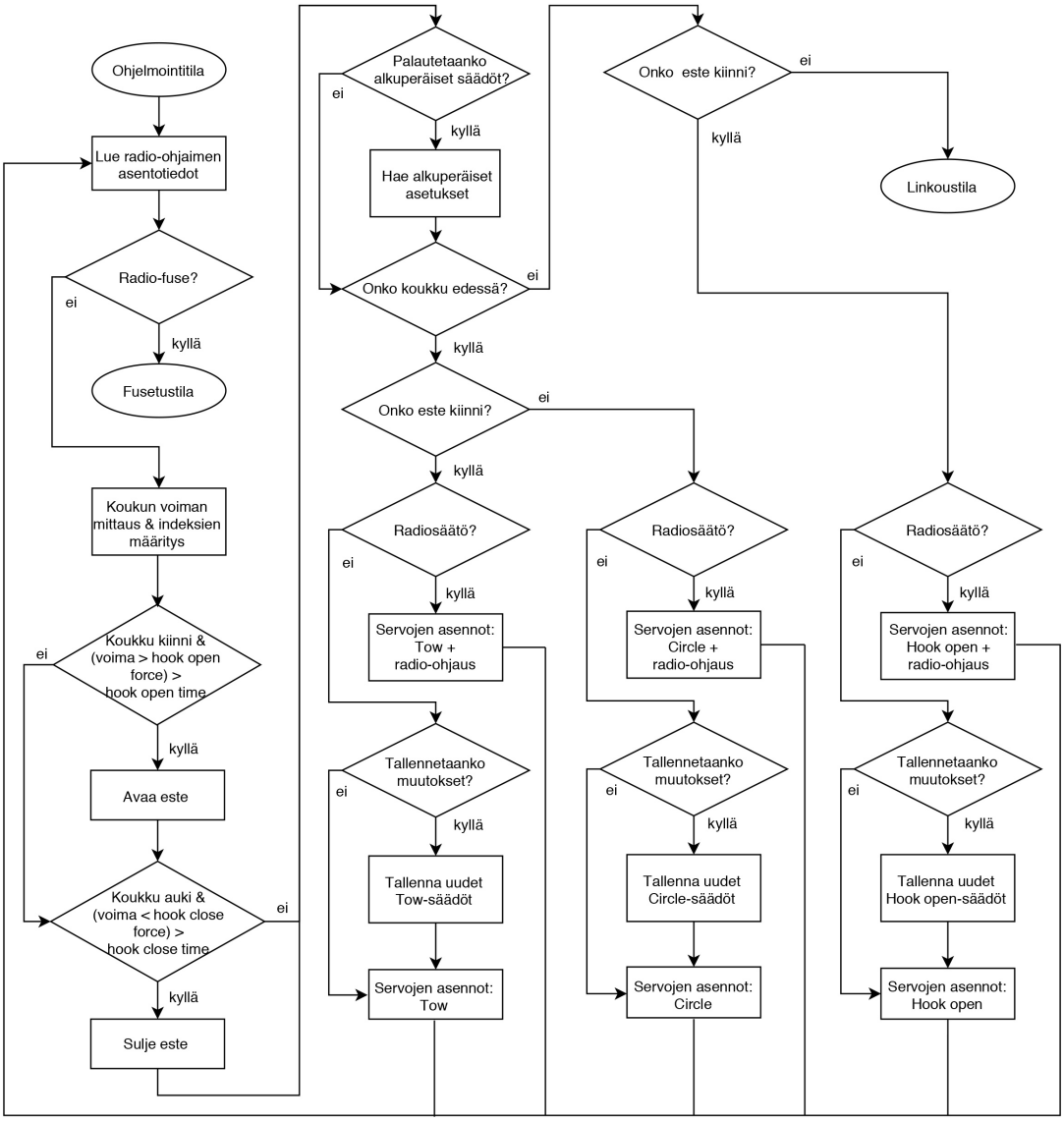
Kuva 16 Hinaustilan lohkokaavio mekaanisella hinauskoukulla

Kuvassa 17 on esitetty hinaustilan lohkokaavio, kun käytössä on servokoukku. Hinaussiiman voiman mittausta suodatetaan ohjelmallisesti viiden näytteen keskiarvottavalla suodattimella, jotta ohjauksesta saataisiin rauhallisempi.

Toiminta on monimutkaisempi kuin mekaanisen koukun tapauksessa, koska joka hinaustilaa varten on viidet eri servojen asennot, joista valitaan hinaussiiman vedon mukaan ne, mitä käytetään. Lisäksi täytyy pystyä tallentamaan radiosäädön vaikutus jokaiselle voimavälille, jolla on oltu radiosäädön ollessa päällä. Radio-ohjauksen poikkeutuksien tallentamista varten on ohjelmassa kolme viisi alkioista taulukkoa, jokaiselle hinaustilalle omansa. Jotta hinaussiiman vetovoimaa vastaavat ohjainsauvojen poikkeamat saadaan tallennettua oikeaan taulukkoon ja oikeaan alkioon, ohjelmassa on

oma indeksimuuttuja jokaiselle taulukolle. Indeksimuuttujien arvot määritetään mitatun voiman perusteella ja samojen indeksimuuttujien arvoja käytetään myös servojen asentoja valittaessa mitatun vetovoiman mukaan.

Toiminta on muuten vastaava kuin mekaanisen koukun tilanteessa, mutta kun radiosäätö kytketään poispäältä, lisätään kyseessä olevan hinaustilan kaikkiin eri vetovoiman mukaisiin säätöihin vastaavat radio-ohjauksen muutokset vastaavasta taulukosta.



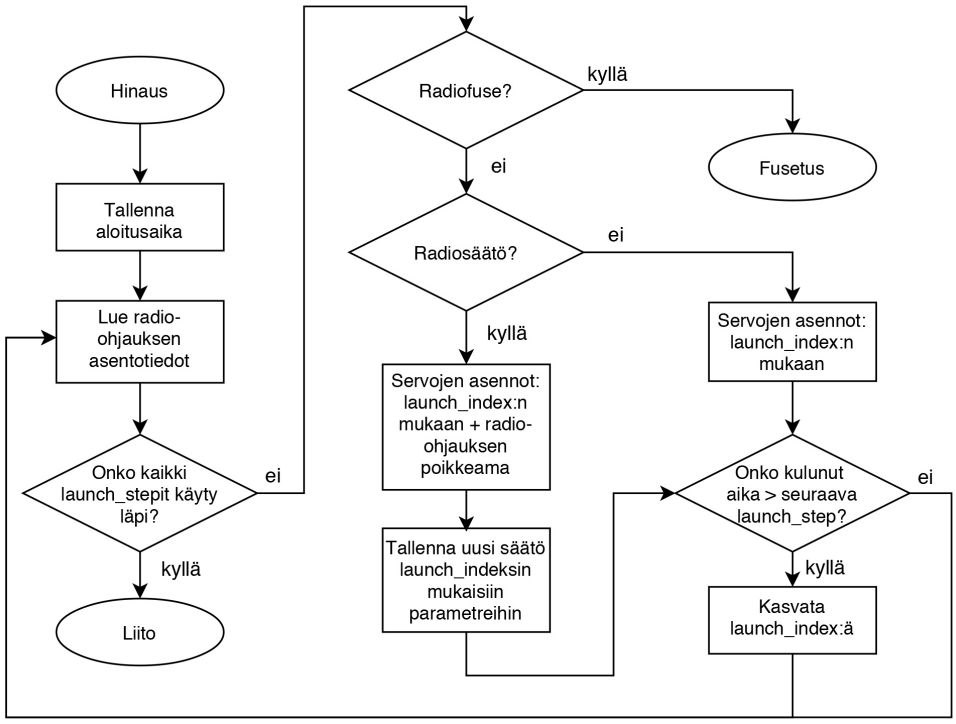
Kuva 17 Hinaustilan lohkokaavio servokoukulla

Kun käytössä on servokoukku, ei tarvita toimintoa, jossa hinaustilasta mennään suoraan liitotilaan, sillä koukun aukeaminen tahattomasti on epätodennäköisempää ja koukku ohjataan takaisin kiinni, kun vetovoima laskee alle asetetun rajan asetetuksi ajaksi koukun ollessa vielä etuasennossa.

Koska radio-ohjaus ei tuota ohjainpintojen absoluuttisia asentoja, vaan ohjainpintojen asennon poikkeamia muistissa oleviin asentoihin, esimerkiksi hinausvaiheessa lennokin tehdessä ympyrää, sivuperäsimen kääntö kaarron suunnalle vastakkaiseen suuntaan, ei välttämättä riitä edes sivuperäsimen kääntämiseksi suoraan. Tarkoitus ei olekaan, että joka tilanteessa lennokkia voidaan ohjata kuten tavanomaista radio-ohjattavaa lennokkia, vaan tarkoitus on tehdä vain hienosäätöjä asetuksiin.

4.1.7 Linkoustila

Linkoustilassa käydään läpi sarja servojen asentoja, joiden määrää ja pituutta voidaan muuttaa. Linkoustilassa radiosäätö toimii vain, jos se on päällä linkoustilaan siirryttäessä, ja se on päällä koko linkouksen ajan, koska linkous on ajallisesti lyhytkestoinen vaihe. Kuvassa 18 on esitetty linkousvaiheen lohkokaavio.



Kuva 18 Linkoustilan lohkokaavio

Linkousvaiheeseen siirryttäessä tallennetaan linkouksen aloitusaika omaan muuttujaansa, jotta myöhemmin voidaan laskea kulunut aika. Linkoustilassa ohjelma kiertää samaa silmukkaa, kunnes koko servojen asentojen sarja on käyty läpi. Servojen asennot sisältävistä taulukoista haetaan arvot indeksin avulla, jota kasvatetaan aina kun aikaa on kulunut asetetun aikavälin verran. Mikäli radiosäätö on päällä lisätään asetettuihin säätöihin ohjainsauvojen poikkeamat keskiasennosta ja tallennetaan muutetut arvot kyseisen aikavälin servoasennot sisältäviin muuttujiin.

Myös linkoustilassa käydään joka kierroksella tarkastamassa, onko tehty radiofusetus, ja jos on niin muutetaan ohjaimen tila fusetustilaan. Linkoustilassa ei tarvitse pystyä palauttamaan alkuperäisiä asetuksia radio-ohjauksen välityksellä, koska radio-ohjauksella muutettuihin asetuksiin ei joka tapauksessa enää palata saman lennon aikana.

4.1.8 Liitotila

Liitotilassa servoilla on vain yksi asento, mihin ne ajetaan liitotilaan siirryttäessä. Radiosäätö toimii siten, että kun radiosäätökytkin laitetaan päälle, lisätään radio-ohjauksen vaikutus servojen asentoon ja kun kytkin käännetään pois päältä, tallennetaan muutetut arvot ohjaimen muistiin. Alkuperäiset asennot voidaan halutessa palauttaa radio-ohjaimen kytkimellä ja myös radiofusetus toimii samalla tavalla kuin muissakin tiloissa. Timer on liitotilassa kunnes asetettu lentoaika on saavutettu tai tehdään radiofusetus. Molemmissa tilanteissa liitotilasta siirrytään fusetustilaan.

Algoritmissa 3 on liitotilan ohjelmakoodin osa, josta nähdään kuinka radio-ohjauksen päälle- ja poiskytkennät sekä säätöjen tallennuksen toteutus on tehty. Kytkimien asennot, joiden perusteella tehdään radiofusetus, alkuperäisten arvojen palautus ja asetetaan radio-ohjaus ja –säätö päälle, lähetetään radiolähettimellä omilla servokanavillaan ja luetaan lennokin ohjaimella taulukosta *Servo_Buffer*, mihin servojen asentotiedot tallennetaan *read_rx*-funktiossa.

Jottei esimerkiksi linkoustilan säätömuutoksia tehtäessä vahingossa muutettaisi myös liitotilan säätöjä, toteutettiin ohjelmaan toiminto, joka edellyttää radio-ohjauksen kytkemistä liitotilaan siirtymisen jälkeen ensin pois päältä ja uudelleen päälle, jotta radio-ohjaus ja –säätö saadaan toimintaan. Muuttuja *radio_tuning* on säätöparametri, joka asetetaan tietokoneohjelmalla, ja jolla voidaan asettaa radiosäätö kokonaan pois päältä. Radio-ohjauksen poikkeamat neutraalista pisteestä luetaan paikallisiin muuttujiin ohjelmakoodin riveillä 25-27 ja kun radio-ohjauskytkin kytkin on käännetty pois päältä, lisätään radiosäädön poikkeamat servojen liitoasentoihin ja asetetaan radiosäädön poikkeamat sisältävät muuttujat nollaksi. Tämä nähdään esimerkkikoodin riveillä 29-36.

Radiosäädön vaikutus kunkin servon asentoon lisätään funktiossa *limit_position*, joka nimensä mukaisesti myös rajoittaa servon liikkeen tietokoneella asetettuihin liikealueen rajoihin, jos radio-ohjauksen ja ohjaimen muistissa olevan säädön summa ylittää raja-arvon. Korkeusperäsimellä käytetään ”työntösuuntaan” erillistä raja-arvoa, jotta vältytään tahattomasta korkeusperäsimen laukeamisesta fusetusasentoon. Työntösuunnalla tarkoitetaan korkeusperäsimen asetuskulman muutosta, joka saadaan aikaan työntämällä ohjaussauvaa, ja joka aiheuttaa lennokin nokan kääntymisen alaspäin. Korkeusperäsimen ohjausmekanismi on toteutettu niin, että mekanismi vapauttaa korkeusperäsimen kun mekanismia poikkeutetaan riittävästi työntösuuntaan, jonka jälkeen kumilenkit vetävät korkeusperäsimen mekaanista rajoitinta vasten n. 45 asteen kulmaan.

```
while (Timer_mode == Glide)
{
  read_rx(); // reads servo positions received from tx
  led_beacon(); //flashes led beacon every 3 second
  if(Servo_buffer[RADIODT] > 1500)
  {
    Timer_mode = Dt;
  } //if load eeprom switch is on, we'll return original servo positions
  if (Servo_buffer[LOADEEPROM] > 1500 && !load_eeprom)
  {
    load_glide_variables();
    load_eeprom = true;
  }
  if (Servo_buffer[LOADEEPROM] < 1500)
  {
    load_eeprom = false;
  }
  if (rx_on_glide_begin && Servo_buffer[RADIOTUNE] < 1500)
  {
    rx_on_glide_begin = false;
  } //If radio tuning is turned on, we'll calculate the differences of
    //the current positions and the neutral positions.
  if (Servo_buffer[RADIOTUNE] >1500 && !rx_on_glide_begin && Radio_tuning)
  {
    ele_rx = Servo_buffer[ELEVATOR]-Ele_neutral;
    rud_rx = Servo_buffer[RUDDER]-Rud_neutral;
    wgg_rx = Servo_buffer[WIGGLER]-Wgg_neutral;
  }
  else //Radio tuning is turned off, differences are added to variables
  {    //containing glide servo positions
    Ele_glide = Ele_glide + ele_rx;
    Rud_glide = Rud_glide + rud_rx;
    Wgg_glide = Wgg_glide + wgg_rx;
    ele_rx = 0;
    rud_rx = 0;
    wgg_rx = 0;
  } //Limiting and setting the positions of the servos
  Servo_position[ELEVATOR] = limit_position(Ele_glide, ele_rx,
                             Ele_servo_lim_pos, Elevator_dt_limit);
  Servo_position[RUDDER] = limit_position(Rud_glide, rud_rx,
                             Rud_servo_lim_pos,Rud_servo_lim_neg);
  Servo_position[WIGGLER] = limit_position(Wgg_glide, wgg_rx,
                             Wgg_servo_lim_pos, Wgg_servo_lim_neg);
  //If flight time is full, timer mode is changed to fuse mode
  if((millis()-Glide_start_time) >= Flight_time*1000)
  {
    Timer_mode = Dt;
  }
}
```

Algoritmi 3 Liitotilan ohjelmakoodi

4.1.9 Fusetustila

Fusetustilassa servojen asennot ajetaan fusetusasentoihin ja parin sekunnin viiveen jälkeen ohjaimen tila muutetaan ohjelmointitilaksi. Fusetustilassa ei ole tarpeellista olla radiosäätöominaisuutta, koska korkeusperäsimen asento rajoitetaan mekaanisella rajoittimella, jota vasten kumilenkit vetävät korkeusperäsimen. Muiden ohjainpintojen asennot eivät ole niin kriittisiä, että niitä tarvitsisi radio-ohjauksella säätää.

4.1.10 Virrankulutus

Lennokin ohjaimen teholähteenä käytettävän akun täytyy olla kevyt ja sen vuoksi kapasiteetiltaan käytännössä hyvin rajallinen. Tästä syystä ohjaimen virrankulutus on tärkeää minimoida varsinkin siinä tilanteessa, kun ohjaimeen on kytketty akku, mutta sitä ei käytetä.

Hyötysuhteeltaan hyvien jänniteregulaattoreiden lisäksi tehonkulutuksen optimoinnin kannalta tärkeää on minimoida virrankulutus sammuttamalla tai laittamalla lepotilaan suuret kuormat silloin kun niitä ei tarvita. Lennokin ohjaimessa ylivoimaisesti suurimman virran ottavat servot, joita tarvitaan ainoastaan lennon aikana. Servoille syötetään käyttöjännite omalla erillisellä jännitehakkurilla, joka voidaan kytkeä mikroprosessorin avulla tarvittaessa päälle tai pois päältä. Ohjaimen ollessa ohjelmointitilassa ei servoille tarvita käyttöjännitteitä, lukuun ottamatta servojen testaustoimintoa. Ohjaimen siirtyessä ohjelmointitilasta hinaustilaan, kytketään servojen käyttöjännitteet päälle ja siirryttäessä takaisin ohjelmointitilaan, kytketään käyttöjännitteet pois päältä.

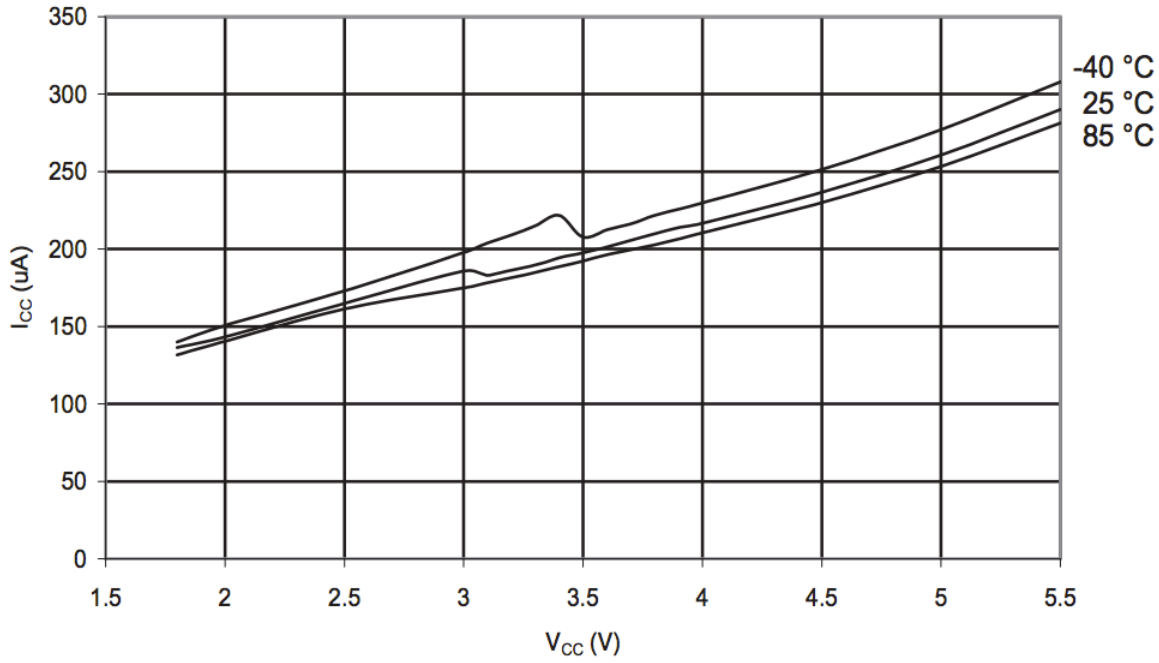
Bluetooth-moduulin virrankulutus on datalehden mukaan 30-40 mA, kun yhteyttä toiseen laitteeseen ei ole ja se aktiivisesti etsii muita bluetooth-laitteita. Bluetooth-yhteyttä ei tarvita muulloin kuin ohjelmointitilassa ja sen päälle- ja poiskytkentä toteutettiin johtamalla sen maadoituspisteet kytkintransistorin läpi maahan.

Radiomoduulinvirrankulutus on aktiivisessa vastaanottotilassa datalehden mukaan 18,5 mA. Se voidaan ohjata ohjelmallisesti eri virransäästötiloihin, joista virrankulutukseltaan pienin on standby-mode. Datalehden mukaan standby-mode:ssa virran kulutus on n. 450 nA.

Ohjaimen ollessa ohjelmointitilassa servojen käyttöjännitteet kytkettynä pois päältä, prosessori, radiomoduuli ja bluetooth-moduuli ottavat yhteensä virtaa muutamia kymmeniä milliampeereja. Jotta turhalta tehon kulutukselta vältyttäisiin, suunniteltiin ohjaimeen virransäästötila, joka aktivoituu timerin ollessa ohjelmointitilassa olematta bluetooth-yhteydessä tietokoneohjelmaan, ja kun hinauskoukku on ollut samaan aikaan taka-asennossa yhtäjaksoisesti yli viisi minuuttia. Virransäästötilassa radiomoduuli ohjataan standby-modeen, bluetooth-moduuli ohjataan pois päältä ja kirjoittamalla

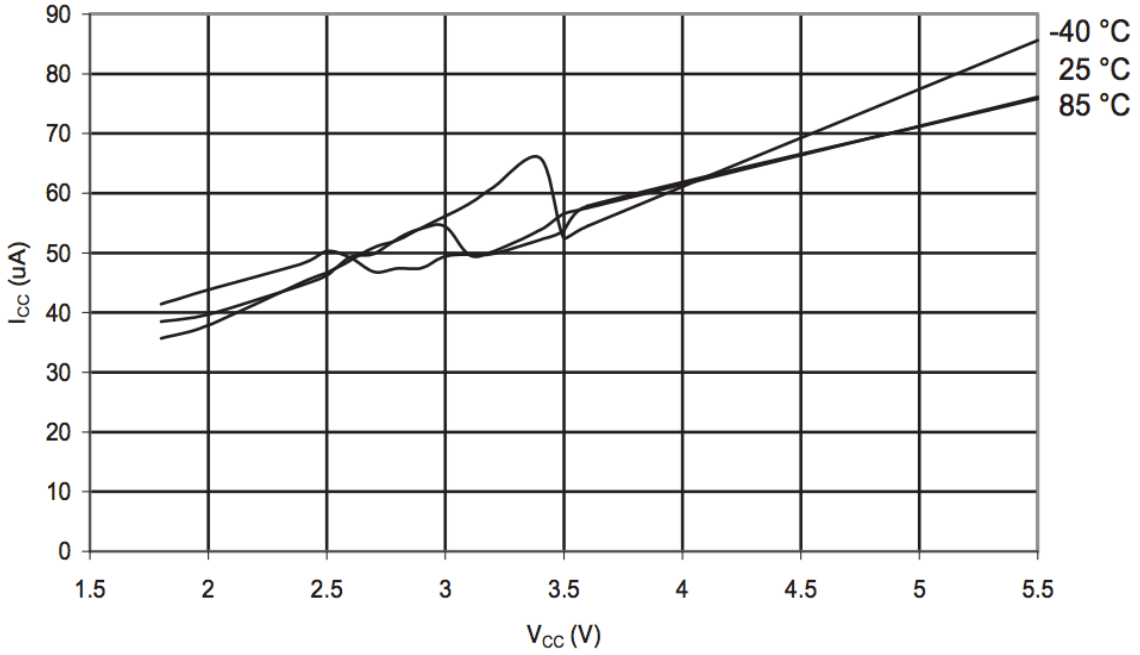
prosessorin PRR-rekisterin (Power Reduction Register) bitit ykkösiksi, prosessorin oheismoduulien kellotus lopetetaan ja näin jäädyttämällä oheismoduulien toiminta, ne eivät kuluta virtaa. Datalehden mukaan sammutettaessa oheismoduulit säilyttävät sen hetkisen tilansa ja herätettäessä ne jatkavat toimintaansa samasta tilasta. Mikroprosessorin PRR-rekisterillä voidaan sammuttaa TWI-moduuli (Two Wire Interface), Timer/Counter0, Timer/Counter1, Timer/Counter2, SPI- ja USART-moduulit sekä ADC-moduuli.

Mikroprosessorin AD-muunnin ja analogiakomparaattori vievät yhteensä suhteellisen suuren lepovirran. Kuvissa 19 ja 20 on Atmega328p:n datalehdellä esitetyt AD-muuntimen ja analogiakomparaattorin virrat käyttöjännitteen funktiona. Nähdään, että käyttöjännitteen ollessa 3,3 V, on AD-muuntimen virta 25°C:n lämpötilassa n. 190 µA ja analogiakomparaattorin virta n. 50 µA. Ennen ADC-moduulin sammuttamista se täytyy asettaa poispäältä ADCSRA-rekisterin (ADC Control and Status Register A) ADEN-bitillä (ADC Enable), koska muuten se vain jäädytetään sen hetkiseen tilaansa ja se jää kuluttamaan sille ominaisen lepovirran.



Kuva 19 AD-muuntimen virta vs. käyttöjännite

Virransäästötilan toteuttava funktio *stand_by* on esitetty algoritmissa 4. Siitä nähdään, että muokattavien rekistereiden arvot tallennetaan muistiin, jotta ne voidaan myöhemmin palauttaa entiselleen. Funktiossa lasketaan prosessorin kellotaajuus 31,25 kHz:iin asettamalla kellotaajuuden esijakaja 1/256:ksi CLKPR-rekisterin (Clock Prescaler Register) avulla. Ennen kuin esijakajan määrääviä bittejä pystytään muuttamaan, täytyy CLKPR-rekisterin bitti CLKPCE (Clock Prescaler Change Enable) kirjoittaa ykköseksi ja tämän jälkeen seuraavan neljän prosessorin syklin aikana voidaan muuttaa esijakajan arvoa.



Kuva 20 Analogiakomparaattorin virta vs. jännite

Virransäästötilasta ohjain herää takaisin ohjelmointitilaan, kun hinauskoukku käytetään etuasennossa. Tällöin *stand_by*-funktiossa muokattujen rekisterien arvot palautetaan entiselleen, nostetaan prosessorin kellotaajuus takaisin 8 MHz:iin, alustetaan SPI-väylä ja sarjaportti sekä herätetään radio-moduuli.

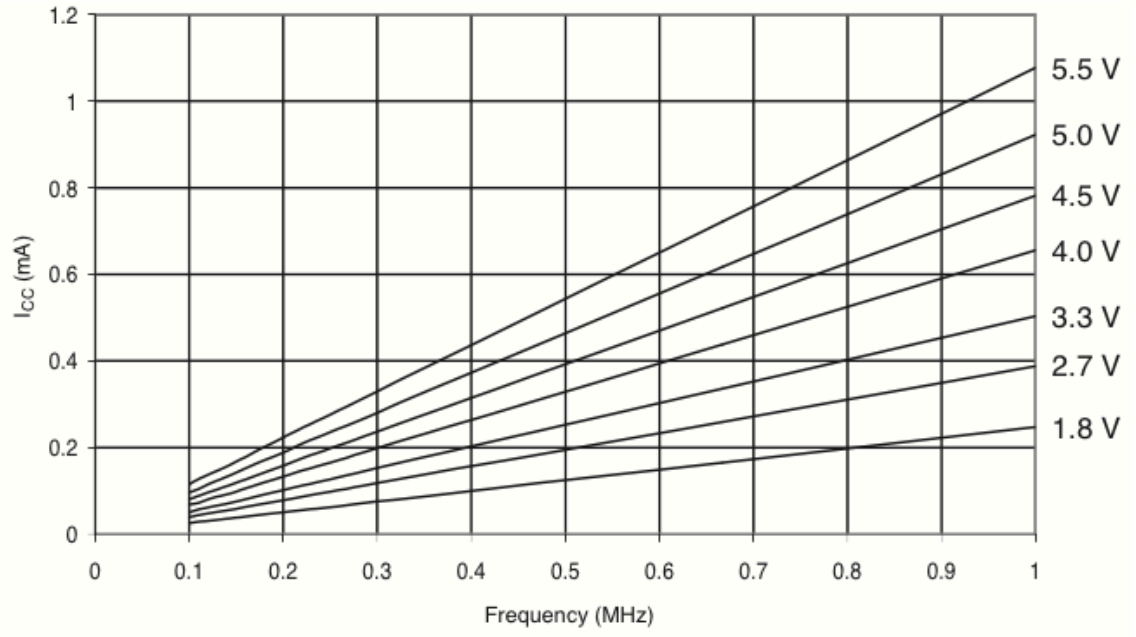
```
1  void stand_by()
2  {
3    digitalWrite(Bluetooth_power_OUT, LOW);
4    digitalWrite(Servo_power_OUT, LOW);
5
6    to_sleep_mode(); // radio module to sleep mode
7
8    Old_adcsra = ADCSRA; // save old adc register
9    Old_prr = PRR; //save old power reduction register
10   ADCSRA = 0x00; // turn off ADC
11   PRR = 0xFF; // turn off peripherals
12
13   cli(); // disable iterrupts because following is timed
14   CLKPR = 0x80; //enable change of CLKPS bits
15   CLKPR = 0x08; //set clock prescaler to 1/256 (8MHz => 31,25kHz)
16 }
```

Algoritmi 4 Funktio stand_by

Kun prosessorin kellotaajuutta lasketaan, laskee myös sen kuluttama virta lähes samassa suhteessa. Kuvassa 21 on Atmega328p:n datalehden esittämä kuvaajaa prosessorin

virrankulutuksesta eri käyttöjännitteillä 0,1 – 1 MHz:n kellotaajuuksilla. Kuvaajasta nähdään, että 100 kHz:n kellotaajuudella ja 3,3 V:n käyttöjännitteellä prosessorin virta on n. 0,05 mA ja vaikka kuvaaja ei ylläkään pienemmille kellotaajuuksille, voidaan olettaa, että 31,25 kHz:n kellotaajuudella virta on vielä tätäkin jonkin verran pienempi.

Virransäästötilasta kerrotaan käyttäjälle vilkuttamalla timerin piirilevyllä olevaa lediä muutaman sekunnin välein. Tämä lisää teoriassa hieman virrankulutusta ledin ottaman virran verran sen ajan kun se palaa. Virran säästötilassa Timer/Counter0 on kytketty pois päältä, joten Arduinon *delay*- tai *millis*-funktioita ei voida käyttää viiveiden suorittamiseen tai laskemiseen vaan viiveiden toteutus tehtiin kasvattamalla viivemuuttujan arvoa joka kierroksella ja nollaamalla se säännöllisesti. Sopivat viiveet ledin vilkutusta varten saatiin aikaan kokeilemalla.



Kuva 21 Atmega328p virrankulutus aktiivisessa tilassa 0,1 – 1 MHz:n kellotaajuudella

Lennokin ohjaimen ollessa lepotilassa sen virta koostuu prosessorin virrasta, radiomoduulin lepovirrasta, akun jännitteenmittauskytkennän ottamasta virrasta, jännitehakkurin häviöistä sekä muista häviöstä ja vuotovirroista. Akkujännitteen mittauskytkennän ottama virta on Ohmin lain mukaisesti n. 0,255 mA, kun akkujännite on 8,4 V ja mittauskytkennän sarjaan kytkettyjen vastuksien kokonaisresistanssi 30 kΩ. Jos häviöitä ei oteta huomioon, lennokin ohjaimen kokonaisvirta lepotilassa pitäisi olla n. 0,3 mA.

Jos lennokin ohjaimen akkujännite laskee alle 7,2 V, ajetaan prosessori lepotilaan, jotta akun tyhjeneminen olisi mahdollisimman hidasta. Lepotilassa prosessorin virran kulutus pitäisi olla datalehden mukaan jopa ainoastaan 0,1 μA 1 MHz:n kellotaajuudella. Lepotilasta ei ole tarvetta herättää timeria enää toimintaan ennen akun vaihtoa.

4.2 Tietokoneen ohjelma

Tietokoneen ohjelman avulla pystytään lukemaan ja kirjoittamaan asetuksia ms-timer lennokin ohjaimeen. Se toteutettiin Processing-nimisessä kehitysympäristössä hyödyntäen ControlP5-nimistä ohjelmakirjastoa.

4.2.1 Processing ja ControlP5

Processing on Java-pohjainen ohjelmointikieli ja vapaan lähdekoodin kehitysympäristö. Processing on alun perin kehitetty tietokoneohjelmoinnin opetusta varten ja se tarjoaa helpon tavan kehittää visuaalisia ohjelmia. Siihen on saatavana lukuisia valmiita ohjelmakirjastoja. Processing:ssa on eri ohjelmointitiloja, joilla on mahdollista tuottaa ohjelmia erilaisille alustoille. Vakio-ohjelmointitiloja ovat Java ja Experimental ja saatavana on esimerkiksi JavaScript ja Android.[16]

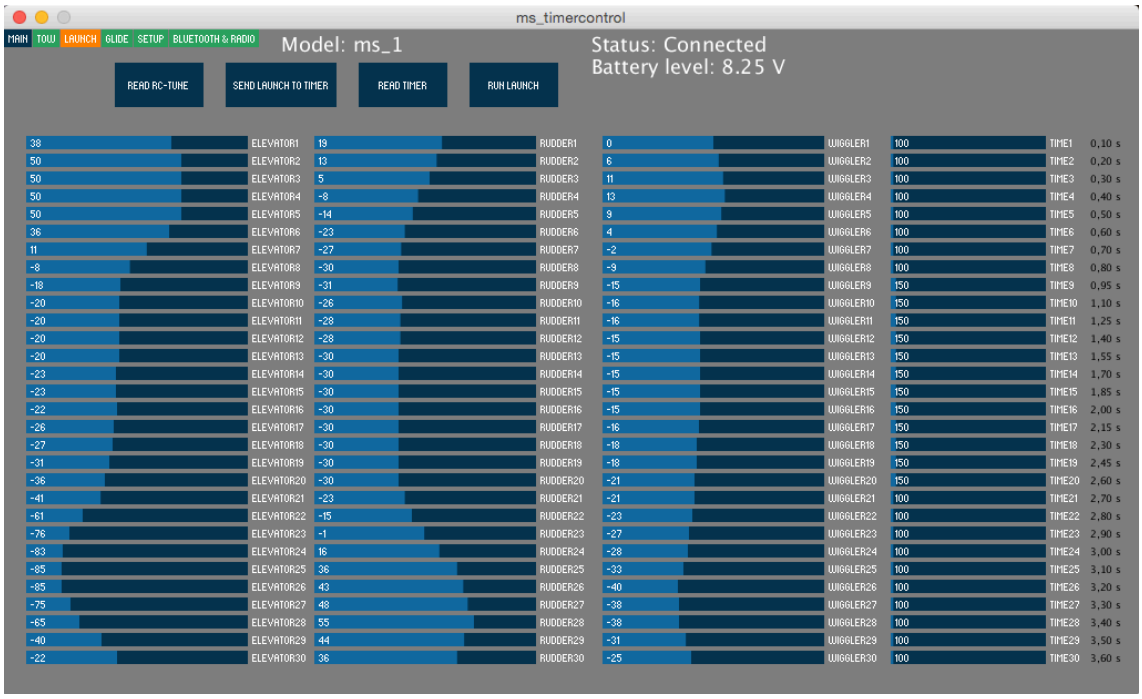
Processing-ohjelmakoodi sisältää kaksi pääfunktiota, *setup* ja *draw*. Funktio *setup* ajetaan kerran ohjelman käynnistyksessä, jonka jälkeen funktiota *draw* toistetaan kunnes ohjelma lopetetaan. Funktio *serialEvent* on Processingin oletusfunktio, joka ajetaan aina kun tulevaa sarjaliikennedataa on tarjolla.

ControlP5 on avoimen lähdekoodin graafisen käyttöliittymän kontrolleriohjelmakirjasto Processing:lle, joka sisältää erilaisia kontrollereita, kuten esimerkiksi painikkeita, liukusäätimiä ja tekstikenttiä. Kontrollereille annetaan yksilölliset nimet ja niiden arvoja voidaan asettaa ja muuttaa ohjelmassa ajonaikaisesti. ControlP5 linkittää automaattisesti kontrollerit saman nimisiin muuttujiin ja funktioihin. Kontrollerit voidaan jakaa eri välilehdille ja ne voidaan piilottaa tai lukita tarpeen mukaan. Muutokset kontrollereissa havaitaan helposti funktiolla *ControlEvent,* joka ajetaan aina kun jokin kontrolleri aktivoidaan. ControlP5 tukee Processing:n Java-ohjelmointitilan lisäksi Android-ohjelmointitilaa. [17]

4.2.2 Tietokoneohjelman rakenne

Tietokoneen ohjelma on jaettu eri välilehdille siten, että *Main*-välilehdellä on tiedostojen tallennus- ja lataustoimintojen lisäksi bluetooth-yhteyden muodostamiseen tarvittavat toiminnot. Kuvasta 22 nähtävällä *Launch*-välilehdellä on linkoustilan asetuksien säätimet ja vastaavasti kuvassa 23 olevalla *Tow*-välilehdellä hinaustilaan vaikuttavien servojen asentojen ja muiden parametrien säätötoiminnot. Liitotilan ja fusetustilan servojen asennot sekä lentoaika säädetään *Glide*-välilehdellä. Servojen liikeratojen perusasetukset tehdään *Setup*-välilehdellä, missä on servojen ääriasentojen sekä keskikohdan säätöjen lisäksi koukun servon kiinni- ja auki asentojen säädöt, sekä testaustoiminnot servojen asennoille. Viimeisellä *Bluetooth*-välilehdellä on bluetooth-laitenimen ja pin-koodin muokkaustoiminnot sekä radiomoduulin taajuushyppely-parametrien asetukseen tarvittavat toiminnot.

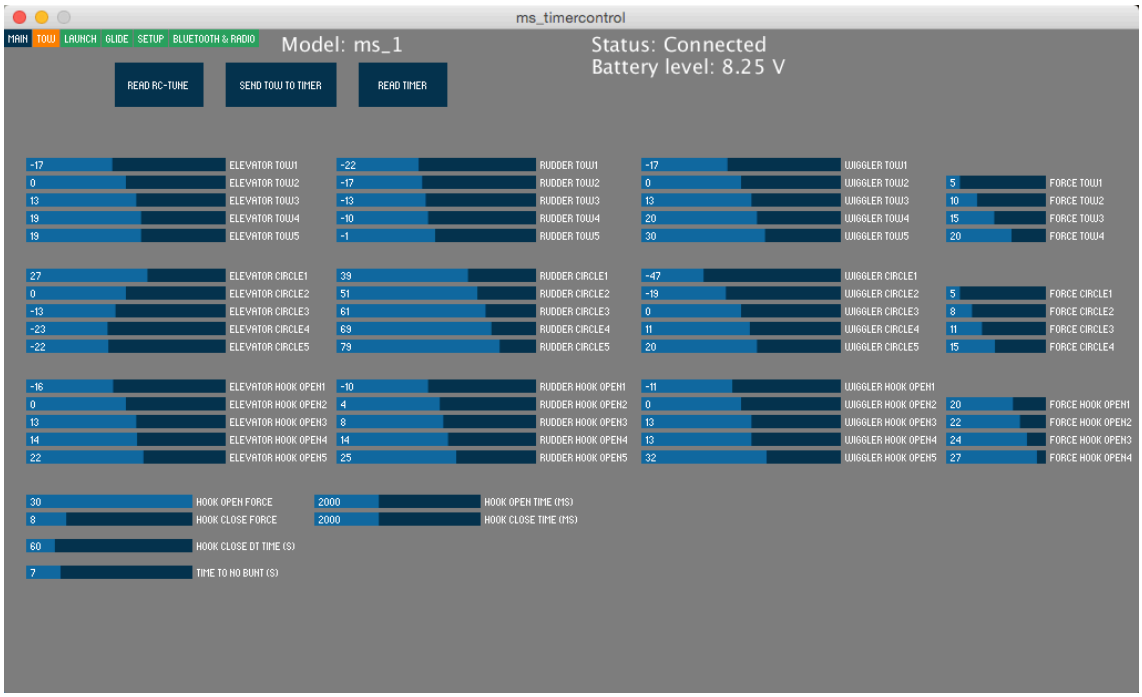
Jokaiselle tietokoneohjelman välilehdelle tehtiin painike, jolla voidaan lukea lennokin ohjaimen EEPROM-muistiin tallennetut parametrit, sekä painike, jolla kyseisen välilehden asetukset kirjoitetaan lennokin ohjaimen EEPROM-muistiin. Main-välilehden kirjoituspainikkeella kirjoitetaan kaikki asetukset lennokin ohjaimen muistiin lukuun ottamatta bluetooth- ja radiomoduulin asetuksia. Lisäksi niillä välilehdillä, joiden asetuksiin on mahdollista tehdä muutoksia radiosäädöllä, tehtiin painike, jolla luetaan lennokin ohjaimen SRAM-muistissa olevat säädöt.



Kuva 22 Näkymä tietokoneohjelman Launch-välilehdeltä

Tietokoneen ohjelmakoodi on jaettu eri tiedostoihin koodin selkeyden parantamiseksi. Tiedosto *timercontrol* sisältää funktiot *setup* ja *draw* sekä funktiot *serialEvent* ja *controlEvent*. Tiedostossa *functions* on yleiskäyttöiset apufunktiot sekä esimerkiksi servojen testausfunktio. Tiedosto *buttons* sisältää ohjelmassa oleviin painikkeisiin linkitetyt funktiot, *file_io*-tiedosto tiedostonkäsittelyä varten tehdyt funktiot ja *serial*-tiedosto sarjaliikenteeseen liittyviä funktioita.

Kaikilla lennokin ohjaimen asetuksilla ei tarvita kaikkia säätöparametreja, joten ylimääräiset säätimet piilotetaan silloin kuin niitä ei tarvita. Esimerkiksi linkousasetuspisteiden lukumäärää voidaan muuttaa ja servokoukun ollessa käytössä on hinausasetuksia paljon enemmän kuin mekaanisella koukulla. Tämän takia toteutettiin näkymän päivittävä funktio *updateVisibleItems*, joka piilottaa tai tuo näkyviin asetettujen säätöjen perusteella tarvittavat kontrollerit. Funktio ajetaan *controlEvent*-funktiossa aina, kun näkymään vaikuttavan asetuksen omaava kontrolleri aktivoidaan. Funktio *controlEvent* saa parametrina aktivoidun kontrollerin nimen ja sen perusteella ohjelma haarautuu eri toimintoihin. Näkymän päivittävä funktio ajetaan myös aina, kun asetuksia luetaan joko tiedostosta tai lennokin ohjaimesta.



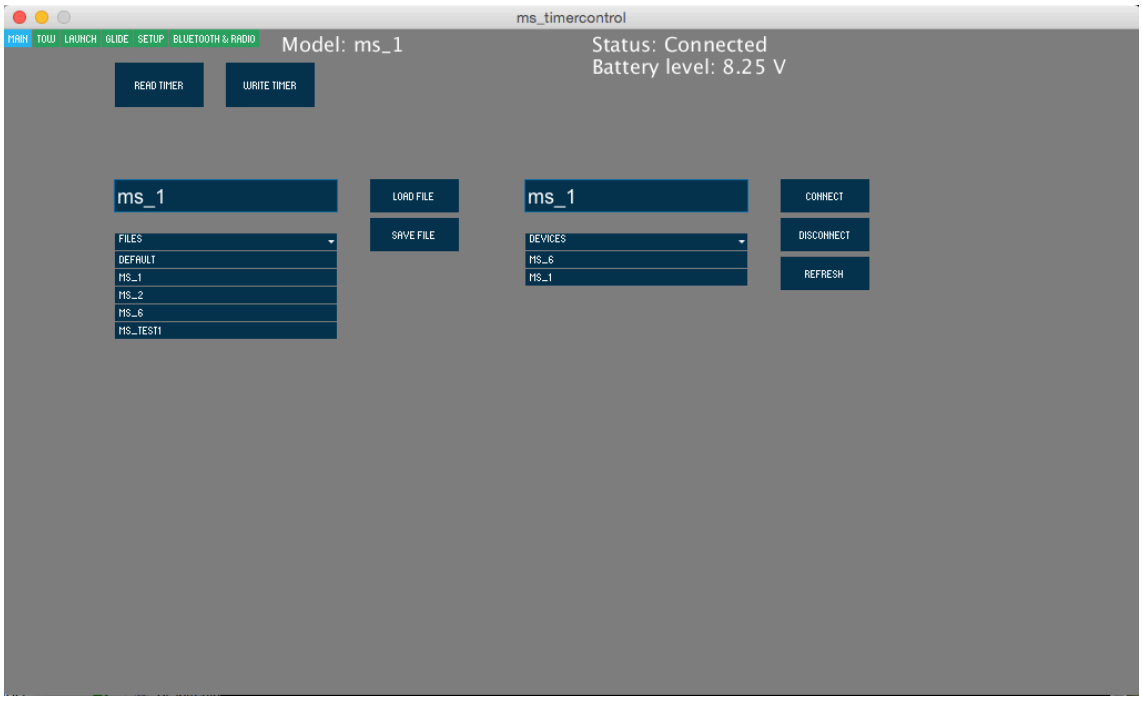
Kuva 23 Näkymä tietokoneohjelman Tow-välilehdeltä

4.2.3 Tiedostonhallinta

Lennokin ohjaimeen tehdyt säädöt ja asetukset voidaan tarvittaessa lukea tietokone-ohjelman avulla ohjaimen muistista, mutta säätöjen varmuuskopioimiseksi on myös tarpeen pystyä tallentamaan säädöt tietokoneelle. Samalle lennokille saatetaan tarvita eri säädöt eri sääolosuhteisiin ja myös tämän takia tietokoneohjelmaan toteutettiin tiedostonlataus- ja tallennustoiminnot.

Processing sisältää vakiokirjastossaan tiedostonkirjoitus funktion *saveStrings* ja tiedostonlukufunktion *loadStrings*, jotka soveltuvat hyvin käytettäväksi säätöjen ja asetusten tallennukseen ja lukemiseen. Funktio *saveStrings* kirjoittaa parametrina annettavan taulukon sisältämät merkkijonot parametrina annettavaan tiedostoon, jokaisen merkkijonon omalle rivilleen. Vastaavasti funktion *loadStrings* lukee parametrina annettavan tiedoston sisältämät merkkijonot taulukkoon, jokaisen tiedoston rivin omaan taulukon alkioonsa.[14]

Tietokoneohjelman *Main*-välilehdelle toteutettiin tiedostojen hallintaa varten toiminnot käyttäen ControlP5-kirjaston Listbox- ja Textfield-luokkia. Ohjelman sijaintipaikassa olevan data-kansion sisältävät .dat-päätteiset tiedostot listataan luetteloksi, josta halutun tiedoston voi valita, tai nimen voi syöttää myös näppäimistöltä tekstikenttään. Tekstikentän sisältämä tiedosto luetaan tai kirjoitetaan save- ja load-painikkeilla. Tietokoneohjelman näkymä *Main*-välilehdeltä on esitetty kuvassa 24.



Kuva 24 Näkymä tietokoneohjelman Main-välilehdeltä, missä ovat tiedostojen lataus- ja tallennus toiminnot sekä sarjaliikenteen yhteyden muodostustoiminnot

Algoritmissa 5 on esitetty tiedoston tallennustoiminnon sisältävä ohjelmakoodi. Lennokin nimi ja tiedoston nimi ovat selkeyden vuoksi aina samat, joten mikäli tekstikentässä on sisältöä, asetetaan koodin rivillä 6 lennokin nimi samaksi kuin tekstikenttään kirjoitettu tiedostonnimi. Tämän jälkeen luodaan taulukko, jonka ensimmäiseen alkioon tallennetaan lennokin nimi ja sen jälkeen kaikki muut asetukset ja säädöt (esimerkkikoodissa vain osa). Kun kaikki arvot ovat taulukossa, kirjoitetaan taulukon sisältö lennokin nimen mukaiseen .dat-päätteiseen tiedostoon ja tallennetaan se data-kansioon.

ControlP5 linkittää automaattisesti kontrollerit saman nimisiin muuttujiin, joten niiden arvo saadaan luettua joko käyttämällä ControlP5-kirjaston funktioita *getController* ja parametrina kontrollerin nimeä merkkijonona sekä funktiota *getValue* tai viittaamalla suoraan muuttujan nimeen. Ensin mainitulla keinolla saadaan kätevästi luettua silmukassa esimerkiksi linkoustilan servojen asennot sisältävien kontrollereiden arvot ja muiden kontrollereiden arvot, joiden nimessä on sama alkuosa ja juokseva numero lopussa. Tämä nähdään esimerkiksi algoritmi 5:n riveiltä 14-18, missä luetaan korkeusperäsimen linkousasetukset taulukkoon.

```
1  void saveFile()
2  {
3    //Textfield filename contains desired name to a outputfile.
4    //Planename will be the same as filename.
5    PlaneName = cp5.get(Textfield.class,"filename").getText();
6
7    if (PlaneName.length() > 0)
8    {
9      int index = 0;
10     String[] data = new String[400]; //Array, where we'll put the data to
11     data[index] = PlaneName;         //be written to the outputfile
12     index++;
13
14     for (int i = 1; i <= LAUNCHPOINTS; i++)
15     {
16       data[index] = str(cp5.getController("elevator"+i).getValue());
17       index++;
18     }
19     for (int i = 1; i <= LAUNCHPOINTS; i++)
20     {
21       data[index] = str(cp5.getController("rudder"+i).getValue());
22       index++;
23     }
24
25     // ...
26     // ...
27
28     data[index] = str(strainCalZero);
29     index++;
30     data[index] = str(strainCalFactor);
31     index++;
32
33     //saveStrings function outputs the content of the data Array to a file
34     //in data-folder.
35     saveStrings("data/"+PlaneName+".dat", data);
36
37     //Next function updates file-listbox. This has to be done because
38     //we might have a new file in data-folder.
39     updateFileList();
40   }
41 }
```

Algoritmi 5 Tiedostontallennuskoodi

Esimerkkikoodin rivillä 31 oleva funktio *updateFileList* tarvitaan päivittämään tiedostoluettelo, jotta mahdollinen uusi tiedosto saadaan näkymään siinä. Tässä hyödynnetään Processing:n oletuskirjaston funktiota *listFileNames*, joka lukee parametrina annetun hakemiston sisältämät tiedostojen nimet merkkijonotaulukkoon. Taulukon alkiot käsitellään yksikerrallaan erotelleen .dat-päätteiset tiedostot ja lisäten

niiden alkuosat tiedostoluetteloon. Funktion *listFileNames* ohjelmakoodi on esitetty algoritmissa 6.

```
1  void updateFileList()
2  {
3    String path = sketchPath + "/data";
4    //An array where we we'll put the filenames of the data-folder
5    //with Arduinos listFileNames-function
6    String[] fileNames = listFileNames(path);
7    //FileIndex contains the number of filenames added in listbox Files
8    FileIndex = 0;
9    //First we'll clear the listbox
10   Files.clear();
11
12   //In the following we'll go through every filename listed in
13   //fileNames-array and if the file ending is .dat, we'll add
14   //the name of the file in the listbox Files
15   for (int i=0; i < fileNames.length; i++)
16   {
17     String fName = "";
18     String fEnd = "";
19     boolean foundDot = false;
20
21     for (int j = 0; j < fileNames[i].length(); j++)
22     {
23       if (fileNames[i].charAt(j) == '.')
24       {
25         foundDot = true;
26         j++;
27       }
28       if (foundDot)
29       {
30         fEnd += fileNames[i].charAt(j);
31       }
32       else
33       {
34         fName += fileNames[i].charAt(j);
35       }
36     }
37     if (!fName.equals("") && fEnd.equals("dat"))
38     {
39     Files.addItem(fName, FileIndex);
40     FileIndex++;
41     }
42   }
43 }
```

Algoritmi 6 Funktio updateFileList

4.2.4 Testi- ja kalibrointitoiminnot

Tietokoneohjelman *Setup*-välilehdelle tehtiin servojen asentojen ja liikeratojen testaamista varten liukusäätimet, joilla voidaan kokeilla missä asennossa mikäkin servo on milläkin asetuksella. Näiden tarkoituksena on helpottaa ja nopeuttaa uuden lennokin säätöjen tekemistä.

Tietokoneen ohjelmassa servojen asennot voidaan määritellä välillä -125…125 yksikköä. Servojen liikealueiden rajat voidaan määrittää erikseen molempiin suuntiin, eli välillä -150..0 ja 0…150. Lisäksi servon keskikohdan säätöä varten on oma säätimensä, jonka avulla servon asentoon lisätään offset välillä -25…25. Halutessa voidaan servon toiminta voidaan kääntää peilikuvaksi reverse-valinnalla. Nämä säätimet nähdään kuvassa 25, jossa on tietokoneohjelman näkymä *Setup*-välilehdeltä.

Kuva 25 Näkymä tietokoneohjelman Setup-välilehdeltä

Servojen asennot muutetaan servopulssin pituudeksi vastaavalla funktiolla kuin aikaisemmin esitetty lennokin ohjaimen funktio *convert_to_us*, mutta nyt laskenta tehdään tietokoneohjelmassa, koska laskennassa täytyy huomioida myös servon liikkeen raja-arvot, keskipisteen säätö ja reverse-valinta, joita kaikkia voidaan muuttaa testauksen aikana.

Kun testaustoiminnan päällekytkevää Test-painiketta painetaan, lähettää tietokone-ohjelma kertaalleen jokaisen servon sen hetkisen ohjauspulssin pituuden lennokin ohjaimelle. Tämän jälkeen lennokin ohjaimelle lähetetään ainoastaan niiden servojen ohjauspulssien pituudet, joiden säätöjä muutetaan. Toteutus tehtiin erillisellä servontestausfunktiolla *testServo*, joka ajetaan, jos *controlEvent*-funktiossa havaitaan,

että jokin servojen asetus- tai testauskontrollereista on aktivoitunut. Funkiolle *testServo* annetaan parametrina servon numero (1: wiggler, 2: korkeusperäsin, 3: sivuperäsin ja 4: koukku).

Hinauskoukun voiman mittausta varten tehtiin tietokoneohjelman *Setup*-välilehdelle kalibrointi-painikkeet, joilla mitataan instrumentointivahvistimelta tuleva analogia-signaali rasittamattomana ja 10 kg painolla rasitettuna. Näistä mittauksista saadaan lennokin ohjaimelle tallennettua offset-virheenkorjausparametri ja korjauskerroin voiman laskemista varten.

4.2.5 Bluetooth-sarjaliikenne

Lennokin ohjaimessa käytettävä bluetooth-moduuli on tarkemmin ilmaistuna bluetooth-sarjaliikennesovitin. Jotta yhteyden muodostaminen siihen onnistuisi helposti tietokoneohjelmasta, tehtiin ohjelmaan toiminto, jolla poimitaan tietokoneen sarjaporttilistauksesta ms timer-laitteet ja tehdään niistä listaus. Listaus tehtiin samaan tapaan kuin tiedostojen listaus, mutta siinä käytettiin Processing:n funktiota *Serial.list*, joka tallentaa olemassa olevien sarjaporttien nimet parametrina annettavaan taulukkoon. Taulukosta etsitään merkkijonon ”ms_timer” sisältävät sarjaportit ja siistitään nimiä hieman ennen listaan lisäystä. Samalla tallennetaan muistiin kyseessä olevan sarjaportin indeksinumero, jolla se myöhemmin löydetään kun yhteyttä muodostetaan.

Sarjaliikenneyhteyden avaamiseen tarvittava ohjelmakoodi on esitetty algoritmissa 7. Jos yhteyden muodostaminen ei onnistu, ohjelma heittää poikkeuksen, jonka käsittely on esimerkkikoodin riveillä 20…26. Siinä ilmoitetaan käyttäjälle, ettei yhteyden muodostus onnistunut, ja nollataan sarjaliikenne, jotta yhteyden muodostamista voidaan yrittää uudelleen.

Sarjaliikenneyhteyden muodostamisen jälkeen kirjoitetaan sarjaporttiin merkkiä ’A’ toistuvasti, kunnes vastauksena saadaan ”ACK” ja *serialEvent*-funktio palauttaa muuttujan *serialFree* arvoksi tosi. Jos vastausta ei saada 5 sekunnin kuluessa, ilmoitetaan käyttäjälle virhetilasta, eikä muita toimintoja suoriteta. Jos vastaus saadaan, voidaan poistaa lukitus painikkeista, joissa käsitellään sarjaliikennettä. Jos sarjaporttiin yritettäisiin kirjoittaa, kun sarjaliikenneyhteys ei ole toiminnassa, ohjelma heittäisi poikkeuksen. Poikkeuksien käsittelyltä vältytään lukitsemalla painikkeet aina kun sarjaliikenneyhteys ei ole toiminnassa.

Datan lähetys molempiin suuntiin tapahtuu kirjoittamalla sarjaporttiin ensin lähetettävien tavujen määrä, sen jälkeen itse data ja lopuksi tarkistussumma. Tietokoneelta timerille päin kirjoittaessa lisäksi kirjoitetaan aina ensin käskytavu, jotta timerin sarjaliikenteen käsittely osaa suorittaa halutut toimenpiteet.

```
1  void connect()
2  {
3    // reset connection
4    if(myPort != null)
5    {
6      myPort.stop();
7      myPort = null;
8    }
9    SerialError = false;
10   SerialStatus = "";
11
12   try
13   {
14     myPort = new Serial(this, Serial.list()[Devices[DeviceIndex]], 57600);
15   }
16   // if the device is unavailable or something like that, we'll end here
17   catch (Exception e)
18   {
19     SerialError = true;
20     SerialStatus = "Device not found!";
21   }
22   if (myPort != null)
23   {
24     int contactTime = millis();
25     // if the device answers, variable Connected will be changed to true
26     // in function serialEvent
27     while (!Connected && !SerialError)
28     {
29       myPort.write('A');
30       delay (100);
31       // if the device doesn't answer in 5 second we'll stop trying
32       if (millis()-contactTime > 5000)
33       {
34         SerialError = true;
35         SerialStatus = "Error, not connected!";
36       }
37     }
38     // if everything goes well, we can unlock buttons that has
39     // serial activity
40     if (!SerialError)
41     {
42       unlockButtons();
43       SerialEstablishTime = millis();
44     }
```

Algoritmi 7 Ohjelmakoodi jossa muodostetaan sarjaliikenneyhteys

Tarkistussumma lasketaan yksinkertaisella CRC-algoritmilla (Cyclic Redundancy Check), joka tuottaa 8-bittisen luvun. Tämä luku lasketaan sekä lähettävässä että vastaanottavassa päässä ja vertaillaan keskenään. Jos tarkistussummat eivät täsmää, pyydetään uudelleenlähetystä. Tarkistussumman lähetyksellä pyritään varmistamaan

tiedon siirtyminen korruptoitumattomana, ja sitä kautta estämään timerin ohjaimen mahdollisesti tuhoisan virheellisen asetuksen tai ajastuksen syntyminen. Käytetty tietokoneohjelman CRC-algoritmi on esitetty algoritmissa 8.

```
 1  int CRC8(int [] data, int len)
 2  {
 3    int index = 0;
 4    int crc = 0;
 5    while (len > 0)
 6    {
 7      int extract = data[index];
 8      index++;
 9
10      for (int j = 0; j < 8; j++)
11      {
12        int sum = (crc ^ extract) & 0x01;
13        crc >>= 1;
14        if (sum > 0)
15        {
16          crc ^= 0x8C;
17        }
18      extract >>= 1;
19      }
20      len--;
21    }
22  }
```

Algoritmi 8 CRC-tarkistussumman laskentaan käytetty algoritmi

4.2.6 Bluetoothin konfigurointi

Bluetooth-laitteelle on pystyttävä antamaan yksilöllinen nimi, jotta sen käyttäjä tietää ottavansa yhteyden omaan laitteeseensa. Jotta omaan laitteeseen ei pystyisi kuka tahansa muodostamaan yhteyttä on bluetooth-laitteissa myös yksilöllisesti asetettavissa oleva pin-koodi. Bluetooth-laitteen nimen ja pin-koodin muuttamiseksi tietokone-ohjelmaan tehtiin omat toiminnot.

Bluetooth-moduulia voidaan käskyttää niin sanotuilla AT-komennoilla, joilla voidaan esimerkiksi asettaa bluetooth-moduulin sarjaliikenteen nopeus, laitteen nimi ja pin-koodi. AT-käskyt alkavat aina kirjainyhdistelmällä AT, jonka jälkeen välittömästi lähetetään itse käsky. HC-06 bluetooth-moduulille ei voida lähettää AT-käskyjä bluetoothin välityksellä vaan käskyt täytyy lähettää sen sarjaliikennenapoihin.

Bluetooth-laitenimen ja pin-koodin vaihtamiseksi tietokoneen ohjelmaan tehtiin molemmille omat funktionsa. Lennokin ohjaimelle lähetetään ensin joko laitteen nimen vaihto- tai pin-koodin vaihtokäsky ja sen jälkeen laitteen nimi tai pin-koodi. Lennokin ohjain lukee käskyn omaan muuttujaansa ja sen jälkeen tulevan nimen tai pin-koodin

taulukkoon. Algoritmissa 9 on esitetty *Serial_read*-funktion osa, joka suoritetaan mikäli sarjaliikennekäskyn sisältävä muuttuja *Serial_command* on arvoltaan *Devicename*. Koodin riveillä 4 – 10 luetaan taulukosta laitteen nimi merkki kerrallaan ja tallennetaan paikalliseen muuttujaan *device_name*. Rivillä 22 oleva funktio *beep* on apufunktio, jolla annetaan summerilla lyhyitä äänimerkkejä parametrina annettu lukumäärä.

Jotta AT-komennot toimisivat, vaatii bluetooth-moduuli sarjaliikenneyhteyden uudelleen avaamisen ja komentojen välille pitkähköt viiveet. Nimen tai pin-koodin vaihdon jälkeen täytyy bluetooth-yhteys avata uudelleen.

```
1  else if (Serial_command == Devicename)
2  {
3    unsigned int index = 0;
4    String device_name = "";
5
6    while (index < Serial_count)
7    {
8      char c = serial_in_array[index];
9      device_name += c;
10     index++;
11   }
12
13   Serial.end();
14   delay(3000);
15   Serial.begin(57600);
16   delay(1000);
17   Serial.print("AT+NAMEms_timer_"+device_name);
18   delay(2000);
19   Serial.end();
20
21   Serial_command = None;
22   beep(3);
23   Serial.begin(57600);
24 }
```

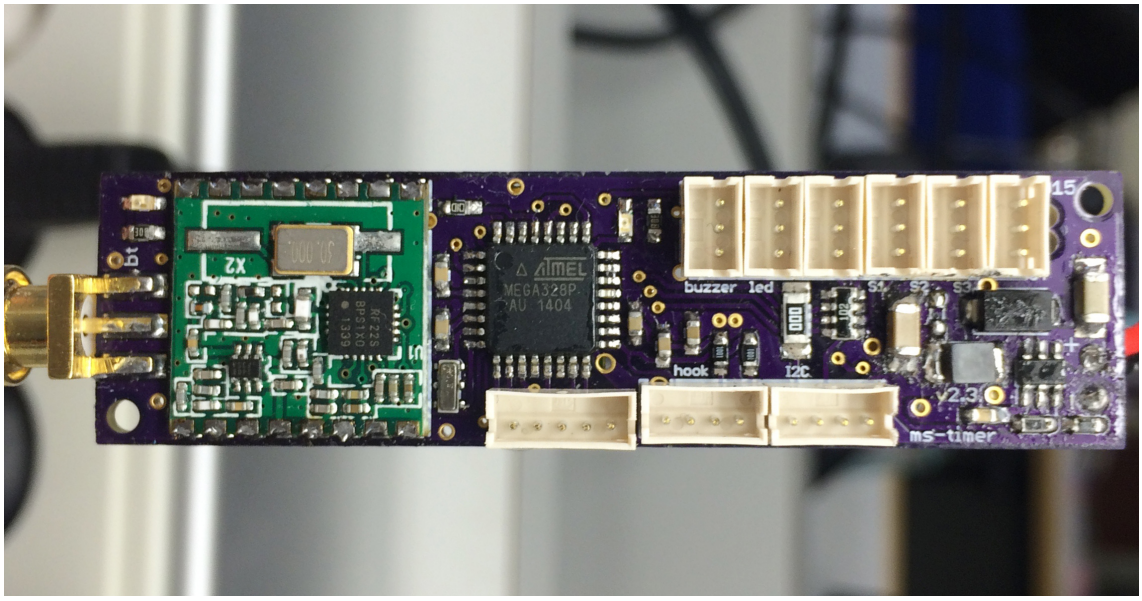
Algoritmi 9 Bluetooth-laitteen nimenmuutoskoodi

5. LOPPUTULOS JA MITTAUKSET

Lennokin ohjain ms-timerin toimivuutta tarkasteltiin käytännön kokeissa ja sen sähköisiä ominaisuuksia tutkittiin erilaisilla mittauksilla.

5.1 Piirilevyn toteutus

Suunnitellut piirilevyaihiot tilattiin OSH Park -nimisestä yrityksestä ja komponentit juotettiin TTY:n elektroniikan laitoksen laboratoriossa. Valmiin piirilevyn mitat olivat n. 64 x 17 mm ja se mahtui helposti koelennoissa käytetyn lennokin rungon yläosaan. Kuvassa 26 on valmiin piirilevyn yläpuoli, vasemmassa reunassa oleva vihreä moduuli on radiomoduuli ja sen vieressä piirilevyn päässä oleva kullan värinen osa on antennin liitin. Yläreunassa olevat 3-napaiset liittimet ovat servojen, summerin ja led-majakan liittimet, alareunassa oleva 5-napainen liitin on ohjelmointiliitin ja sen vieressä ovat hinauskoukun liitin ja I2C-väyläliitin.



Kuva 26 Valmis piirilevy

Kuvassa 27 on ms-timer asennettuna lennokin runkoon, siinä näkyy myös kyseisessä lennokissa käytetty mekaaninen hinauskoukku sekä servot, joiden kiinnityksessä on käytetty keltaista kevlar-punosta.

Kuva 27 ms-timer asennettuna F1A-liidokin runkoon

5.2 Toimivuus

Ms-timer lennokin ohjaimella suoritettiin ensimmäiset koelennot keväällä 2015, ja ensilentojen perusteella toiminnallisuus täytti asetetut vaatimukset. Liidokissa, jolla koelennot tehtiin oli mekaanisella hinaussiiman esteellä varustettu hinauskoukku. Hinaussiiman voiman mittausta ja sen perusteella suoritettavaa lennokin säätöä koestettiin testipenkissä, ja myös sen osalta kaikki toiminnot toimivat suunnitellusti. Servokoukulla varustetulla liidokilla ei pystytty tekemään koelentoja, koska sopivaa liidokkia ei ollut vielä käytettävissä.

Yksi tietokoneohjelman suunnittelun lähtökohta oli se, että uuteen lennokkiin on pystyttävä tekemään perusasetukset mahdollisimman helposti ja niitä on pystyttävä myös helposti testaamaan. Tietokoneohjelmaan toteutettu servojen testaustoiminto osoittautuikin erittäin toimivaksi ja käytännölliseksi. Servon liikealueen rajat haetaan kohdalleen asettamalla servon liike testausliukusäätimestä aivan maksimiin jompaankumpaan päähän ja sen jälkeen kasvatetaan liikealueen raja-arvoa kunnes servo on halutussa maksimiasennossa. Sama toistetaan liikealueen toiselle ääripäälle. Tämän jälkeen voidaan testausliukusäätimellä hakea karkeasti esisäätöjä kohdalleen silmämääräisesti arvioimalla tai mittaamalla ohjainpinnan haluttua asentoa ja sen jälkeen kopioimalla testausliukusäätimen arvo varsinaiseen servon asetusarvoon.

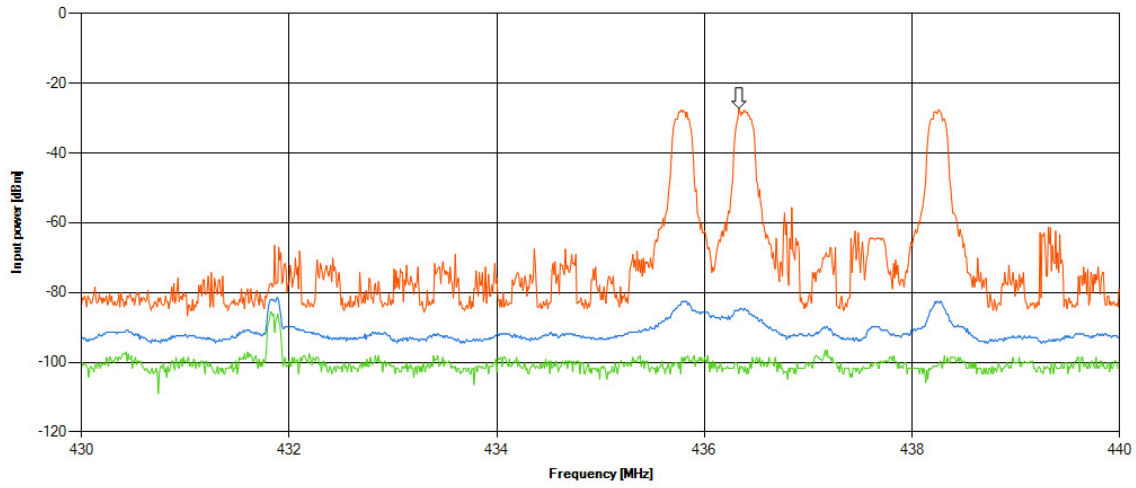
Kannettavan tietokoneen käyttö kenttäolosuhteissa saattaa olla herkälle elektroniikalle liian vaativaa, ja tästä syystä ohjelmisto pitäisi muuntaa jollekin helpommin mukana kannettavalle, ja paremmin vaativia olosuhteita kestävälle alustalle. Matkapuhelimissa

on keskimäärin hieman liian pieni näyttö, mutta minikokoiset tablettitietokoneet soveltuvat tarkoitukseen jo aivan hyvin. Processing:in Android tuki yhdessä Android-ohjelmistokehityksen matalan aloituskynnyksen kanssa tulevat varmasti ohjaamaan valittavaa alustaa.

5.3 Radioantenni ja kantomatka

OpenLRS-ohjelmistoon on kehitetty Windowsissa toimiva Spectrum analyzer -ohjelma, jolla voidaan mitata Rfm22b-radiomoduulilla vastaanotettavan signaalin voimakkuutta. Signaalien mittauksia tehtiin 430 - 440 MHz:n taajuusalueella 10 kHz:n resoluutiolla. Mittausohjelmassa on keskiarvoistava ohjelmallinen suodatin, jonka laskentaan käytettävien näytteiden määrää voidaan säätää. Mittauksen tarkoituksena ei ole löytää terävimpiä tehopiikkejä vaan lähinnä tarkastella yleisesti antennin herkkyyttä mittausalueella. Säädettäessä näytteiden määrää suuremmaksi saadaan siistimmän näköisiä käyriä, mutta mittausnopeus laskee mitä enemmän näytteitä otetaan jokaiselta mitattavalta taajuudelta. Näytteiden määrän asetuksella 2500 saatiin kohtalaisen siistejä mittauskäyriä, mittausnopeuden ollessa vielä kohtuullinen.

Ohuesta teräslangasta rakennetulle maatasoantennille tehtiin mittauksia siten, että radiolähetin asetettiin noin 20 metrin päähän vastaanottimesta. Vastaanotin oli kiinnitettynä telineeseen siten, että antenni osoitti ylöspäin. Ensin tehtiin mittaus tehdasvalmisteisella antennilla, johon itse valmistettua antennia voitaisiin verrata. Ohuesta teräslangasta valmistetulle antennille tehtiin mittaukset siten, että aluksi antennin pituus oli 180 mm. Mittauksia toistettiin ja niiden välillä antennia lyhennettiin n. 1 mm kerrallaan, kunnes sen pituus oli enää 160 mm.



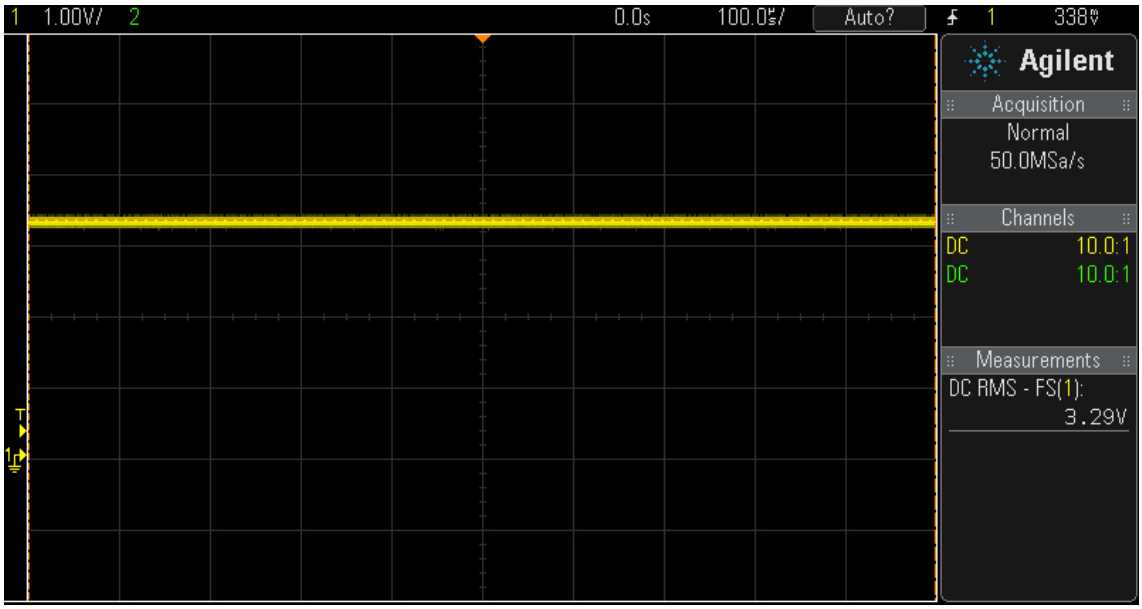
Kuva 28 Käytetyn vastaanotinantennin taajuusspektri

Mittaustuloksista havaittiin, että vastaanotetun signaalin sekä huippu- että keskiarvotasot n. 435 MHz:n taajuudella kasvoivat kun antennin pituus lähestyi 1/4 – aallon mittaa (n. 172 mm) ja laskivat taas kun antennin pituutta lyhennettiin sitä lyhyemmäksi. Kuvassa 28 on esitetty mitatut signaalivahvuudet 172 mm mittaisella antennilla, jolla saavutettiin korkeimmat lukemat. Mittausten perusteella päästiin lähes samaan signaalitasoon kuin tehdasvalmisteisella antennilla. Kuvasta nähdään myös hyvin piikkeinä radiolähettimen taajuushyppelyalueet.

Radion kantomatkaa testattiin radiolähettimen ja -vastaanottimen ollessa maassa. Radio-ohjaus toimi luotettavasti n. 500 metrin päähän. Lennokin ollessa ilmassa, kantomatka todennäköisesti on jonkun verran pidempi, mutta melko varmasti ei kuitenkaan riittävä, sillä ohjauksen pitäisi toimia muutaman kilometrin päähän asti. Radio-ohjauksella tapahtuva säätö tapahtuu käytännössä lennokin ollessa vielä hyvin lähellä, mutta kantomatkan kannalta kriittinen toiminto on radiofusetus, joka jossain olosuhteissa voi olla tarpeellista tehdä, vaikka lennokki olisi kaukanakin.

5.4 Käyttöjännitteet ja virrankulutus

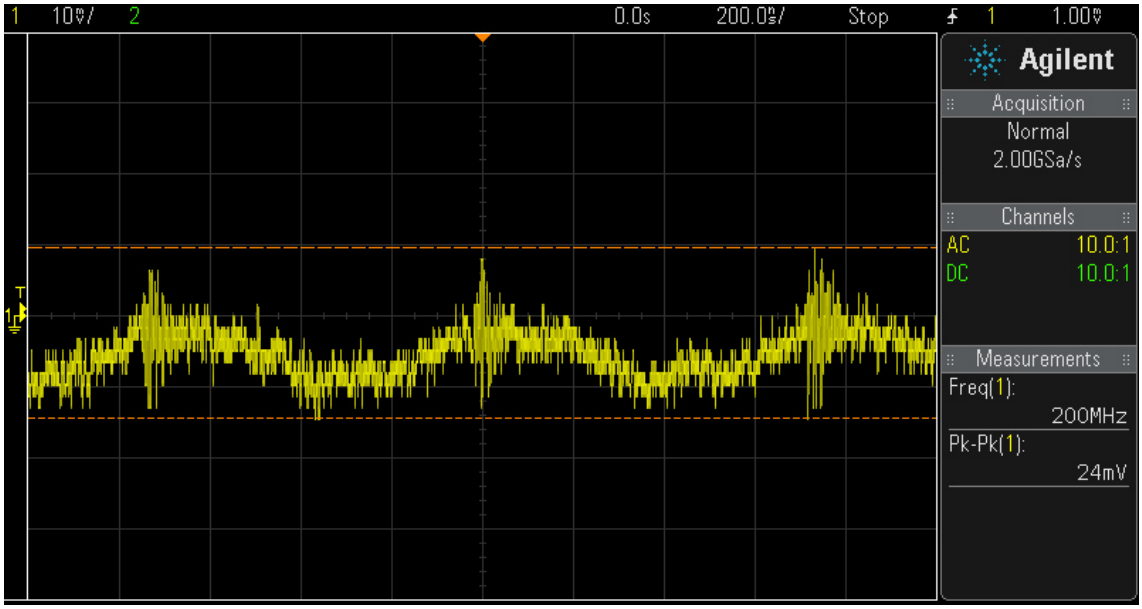
Lennokin ohjaimen käyttöjännitteiden regulointiin suunniteltujen hakkuriteholähteiden ulostulojännitteiden aaltomuodot mitattiin oskilloskoopilla. Kuvassa 30 on 3,3 V:n hakkurin DC-jännite mitattuna oskilloskoopilla. Kuvasta nähdään, että hakkurin ulostulojännite on riittävällä tarkkuudella oikean suuruinen.



Kuva 19 3,3 V:n hakkurin DC-jännite

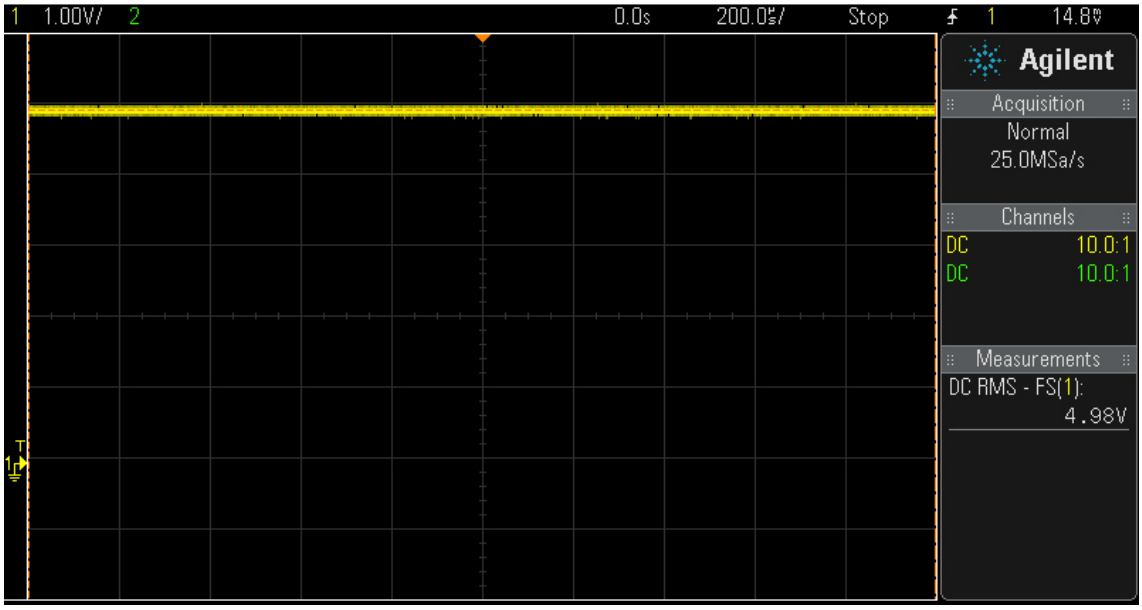
Hakkurin ulostulon jännitteen aaltomuotoa mitattaessa havaittiin, että sen peak-to-peak jänniterippeli on n. 25 mV, mikä on moninkertainen laskettuun teoreettiseen arvoon nähden. Tähän vaikuttavia asioita ovat piirilevyn layout sekä valitut komponentit. Tulosta voidaan kuitenkin pitää vielä kyseessä olevalle sovellukselle välttävänä tasona.

Mittaustarkkuutta heikentää myös mittauksessa esiintynyt n. 10 mV:n yleiskohina. Jännitteen aaltomuoto nähdään kuvasta 31.

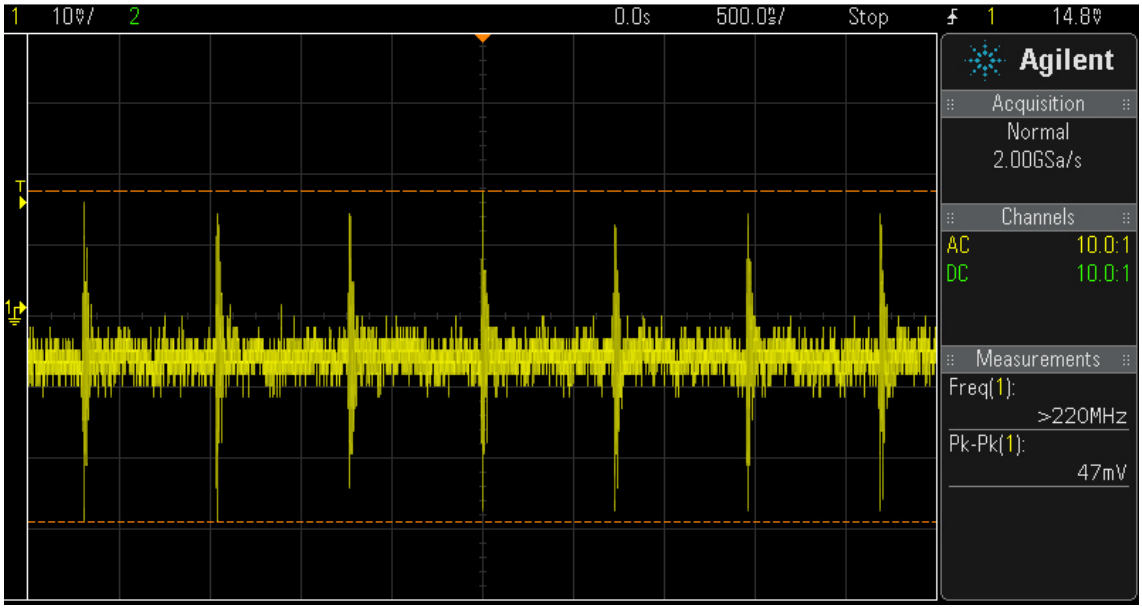


Kuva 31 3,3 V:n jännitehakkurin aaltomuoto

Servojen käyttöjännitteen syöttävälle jännitehakkurille tehtiin vastaavat mittaukset, kuvasta 32 nähdään sen DC-jännite, joka myös vastasi odotuksia riittävän hyvin. Jännitteen aaltomuoto on esitetty kuvassa 33 ja siitä nähdään, että myös 5 V:n hakkurin jänniterippeli on suurempi kuin laskelmissa saatu arvo, mutta myös sitä voidaan pitää vielä välttävänä tasona kyseessä olevalle sovellukselle.



Kuva 32 5 V:n jännitehakkurin DC-jännite

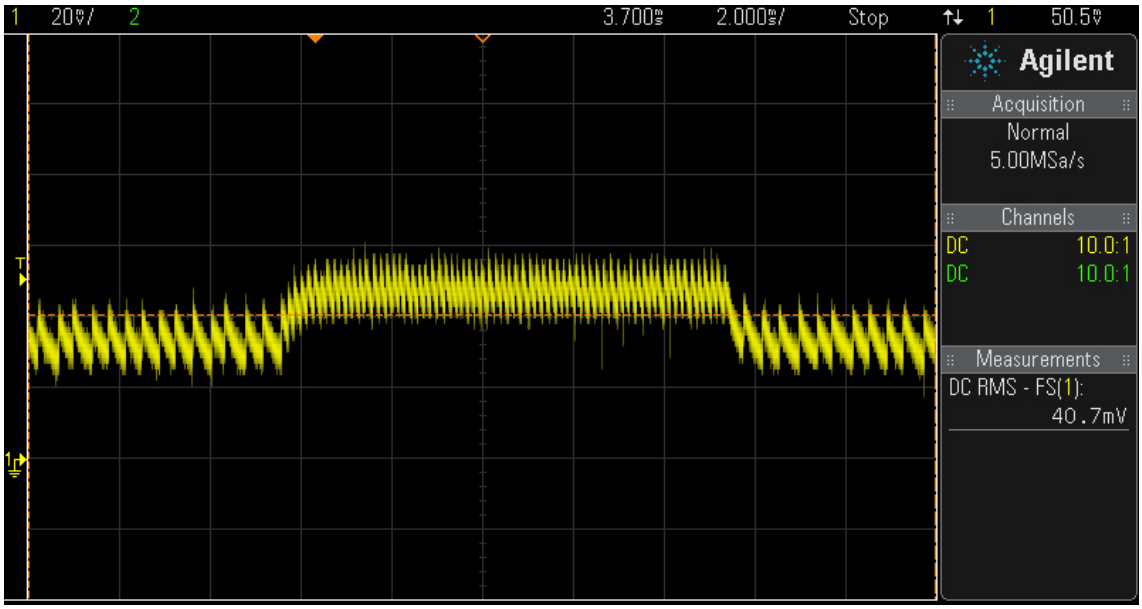


Kuva 33 5 V:n jännitehakkurin aaltomuoto

Lennokin ohjaimen ottama virta mitattiin eri tilanteissa. Ohjelmointitilassa virran suuruus riippuu bluetooth-moduulin tilasta, etsiikö se yhteyttä, vai onko se muodostanut parin toisen bluetooth-laitteen kanssa. Bluetooth-moduulin etsiessä yhteyttä, lennokin ohjaimen virta vaihtelee jaksottaisesti välillä 15-33 mA ja kun bluetooth-pari on muodostettu, timerin virta on n. 26-28 mA.

Kun timer siirtyy ohjelmointitilasta hinaustilaan ja muihin lentotiloihin, bluetooth-moduuli kytketään pois päältä ja servojen käyttöjännitteet kytketään päälle. Lennokin ohjaimen ottama virta ilman servoja lentotiloissa on n. 14 mA. Servojen virrankulutus riippuu servojen tyypistä ja niiden hetkittäisestä mekaanisesta kuormituksesta, joten mittaukset tehtiin ilman servoja.

Jos lennokin ohjain on ohjelmointitilassa pitkään käyttämättömänä, se asetetaan virransäästötilaan, jossa laskennallinen virrankulutus ilman häviöitä on n. 0,3 mA. Mittauksissa huomattiin virran olevan kuitenkin n. 1,1 mA, ja tarkemman selvityksen jälkeen havaittiin, että suurin kuorma on hakkuriteholähde itse, jonka tyhjäkäyntivirraksi mitattiin n. 0,76 mA. Seuraavaksi suurin kuormitus on akkujännitteen mittaukseen käytetyn vastuskytkennän ottama virta. Virransäästötilassa timerin piirilevylle olevaa lediä vilkutetaan muutaman sekunnin välein, jolloin timerin ottama virta kasvaa hetkellisesti. Kuvassa 34 on oskilloskoopilla mitattu 33,2 Ω:n shunttivastuksen yli mitattu jännite. Siitä nähdään, että ledin palamisaika on n. 10 ms ja jännitteen keskiarvo ledin palaessa n. 50 mV, jolloin Ohmin lain perusteella saadaan ohjaimen virraksi ledin palaessa n. 1,5 mA. Käytännössä tämä ei juuri vaikuta kokonaisvirrankulutukseen, koska led välähtää vain hyvin lyhyen ajan muutaman sekunnin välein. Jos akkujännite laskee alle 7,2 V:n asetetaan timer sammutustilaan. Sammutustilassa lennokin ohjaimen virraksi mitattiin myös n. 1,1 mA.



Kuva 34 Mittausvastuksen yli oleva jännite virransäästötilassa ledin vilkutuksen aikana

5.5 Jatkokehitys

Suurin testauksissa todettu toiminnallinen puute ms-timerissä oli radio-ohjauksen liian lyhyt kantomatka. Kantomatkaan vaikuttavista tekijöistä olennaisimmat ovat lähettimen ja vastaanottimen antennit, joissa molemmissa on luultavasti parantamisen varaa, sillä antennien valintaan ja suunnitteluun ei tässä diplomityössä paneuduttu kovin syvällisesti. Jatkossa tullaan testaamaan erityyppisiä antenneja sekä lähettimeen, että vastaanottimeen. Parempien antennien lisäksi ms-timerille tullaan kehittämään radiofuse-lähetin, jonka käyttö myös kilpailuissa on mahdollista. Radiofuse-lähettimessä ei tarvita virtakytkimen lisäksi muuta kuin yksi nappi, jolla fusetus-toiminto laukaistaan. Lisäksi lähettimen täytyy on pieni ja kevyt, jotta sen käyttö liidokin hinauksen aikana olisi mahdollista.

Toinen jatkokehitystä vaativa paikka on käyttöjännitteiden regulointi, koska nyt käytetty 3,3 V:n jännitehakkuri osoittautui pienillä kuormilla hyötysuhteeltaan niin huonoksi, että kulutti itse enemmän virtaa kuin tuotti. Yksi vaihtoehto olisi suunnitella mikroprosessorille oma käyttöjännitteen syöttö pienellä lineaariregulaattorilla, jolloin hakkurijännitelähteen voisi kytkeä pois päältä silloin kun oheislaitteita ei tarvita. Virransäästötilassa myös akkujännitteen mittaukseen käytetyn vastuskytkennän ottama virta oli suhteellisen suuri. Vastuksien kokoa ei voida kasvattaa, koska AD-muunnos ei toimi suuremmilla vastuksilla enää luotettavasti. Mikäli virransäästötilan virta halutaan mahdollisimman pieneksi, täytyisi myös akkujännitteen mittaus toteuttaa toisella tapaa.

Toinen vaihtoehto saattaisi olla servojen käyttöjännitteestä tinkiminen yksikennoisen LiPo-akun jännitteen suuruiseksi (n. 3,7 – 4,2 V), jolloin servoja voitaisiin syöttää suoraan akkujännitteellä ja 3,3 V:n käyttöjännitteen regulointiin pystyttäisiin käyttämään lineaariregulaattoria ilman kohtuutonta tehohäviötä. Yksikennoisen akun jännitteelle optimoituja servoja on tullut markkinoille.

Muita jatkokehityksen kohteita ovat I2C-väylään liitettävät lisälaitteet, kuten esimerkiksi lentokorkeutta mittaava ja tallentava loggeri. Myös telemetria-ominaisuuksia voisi jatkossa kehittää, jolloin lennokista olisi mahdollista saada lennonaikaisia tilatietoja.

6. YHTEENVETO

Diplomityön tavoitteena oli suunnitella ja toteuttaa F1A-liidokin ohjausjärjestelmä, ms-timer, jolla vapaastilentävän lennokin säätö olisi mahdollista perinteistä poikkeavasti lennon aikana, käyttäen radio-ohjainta. Ohjausjärjestelmän säätämistä varten tarvittiin myös tietokoneohjelma, jolla ohjaimen asetuksia pystyttäisiin lukemaan ja kirjoittamaan sekä tallentamaan tietokoneelle.

Työssä päästiin tavoitteeseen niiltä osin, että toimiva lennokin ohjain ja tietokone-ohjelmisto saatiin toteutettua. Käytännön testauksissa ms-timer toimi suunnitellusti ja toiminta vaikutti käyttökelpoiselta ja potentiaaliselta. Jatkossa ohjainta tullaan testaamaan lisää, jolloin testauksen kohteena tulee olemaan myös hinaussiiman vedon mittauksella varustetun servotoimisen hinauskoukun toiminta ja todellinen kantomatka lennokin ollessa ilmassa. Tietokoneohjelma on tarkoitus siirtää Android-pohjaiselle alustalle, jolloin sen käyttö myös vaativimmissa olosuhteissa mahdollistuu.

Suurin puute oli radio-ohjauksen liian lyhyt kantomatka. Tähän vaikuttavat sekä lähettimen että vastaanottimen antennit, joissa on todennäköisesti parantamisen varaa kummassakin. Tämän diplomityön kannalta antennien valinta ja suunnittelu olivat hieman sivuroolissa, mutta jatkossa niihin tullaan keskittymään enemmän.

Kehitettävää jäi myös käyttöjännitteiden regulointiin, koska osoittautui että 3,3 V:n jännitteen regulointiin käytetyn step-down jännitehakkurin hyötysuhde romahtaa, kun kuormitus laskee alle milliampeerin. Virransäästötilassa olevan ms-timerin suurin kuorma on hakkuri itse. Myös akkujännitteen mittaukseen käytetyn vastuskytkennän virrankulutusta on liian suuri, joten myös sille täytyy pohtia muita ratkaisuja.

Jatkokehityksen kohteena on myös sääntöjen sallima radiofuse-lähetin, jota pystyy käyttämään myös kilpailuissa, missä tavallisen lennokkiradion käyttö ei käytännössä ole mahdollista. Muita kehityskohteita ovat telemetriatoiminnot sekä I2C-väylään liitettävät lisälaitteet. Diplomityössä keskityttiin F1A-luokan liidokin tarvitsemiin ominaisuuksiin, mutta jatkossa ms-timerin käyttökohteita voidaan laajentaa myös muihin lennokki-luokkiin.

LÄHTEET

[1] Suomen Ilmailuliitto: Lennokit, vapaastilentävät, verkkosivu
Saatavissa: (viitattu 23.12.2014)
http://www.ilmailuliitto.fi/fi/lajit/lennokit/vapaastilentavat

[2] FAI Sporting Code: Volume F1 Free Flight Model Aircraft, Fédération Aéronautique Internationale, 2014 Edition, 68 p.
Saatavissa: http://www.fai.org/downloads/ciam/SC4_Vol_F1_FreeFlight_2014

[3] AVR910: In-System Programming, 8-bit AVR RISC Microcontroller, Application Note, Atmel Corporation, Rev. 0943E-AVR-08/08, 12 p.
Saatavissa: http://www.atmel.com/images/doc0943.pdf

[4] UM10204 I^2C-bus specification and user manual, NXP Semiconductors, Rev. 6 – 4 April 2014, 64 p.
Saatavissa: http://www.nxp.com/documents/user_manual/UM10204.pdf

[5] N. Mohan, T.M. Undeland, W.P. Robbins, Power Electronics: Converters, Applications, and Design, Third Edition, John Wiley & Sons, Inc., 2003, pp. 164-172

[6] ADP2300/2301, 1.2A, 20V, 700 kHz/1.4 MHz, Nonsynchronous Step-Down Regulator, Datasheet, Analog Devices, Rev. C, 28 p.
Saatavissa: http://www.analog.com/media/en/technical-documentation/data-sheets/ADP2300_2301.pdf

[7] ADP2302/ADP2303, 2A/3A, 20V, 700 kHz, Nonsynchronous Step-Down Regulators, Datasheet, Analog Devices, Rev. A, 28 p.
Saatavissa: http://www.analog.com/media/en/technical-documentation/data-sheets/ADP2302_2303.pdf

[8] Atmega48PA/88PA/169PA/328P: 8-bit AVR Microcontroller with 4/8/16/32K Bytes In-system Programmable Flash, Datasheet, Atmel Corporation, Rev. 8161D-AVR-10/09, 440 p.
Saatavissa: http://www.atmel.com/images/doc8161.pdf

[9] J.J. Carr, Designer's Handbook of Instrumentation and Control Circuits, Academic Press, Inc., San Diego, 1991, pp. 91-96

[10] HC-06 Datasheet, Guangzhou HC Information Technology Co, Ltd, Rev 2.0, 17 p.
Saatavana: http://www.electronicoscaldas.com/datasheet/HC-06_Wavesen.pdf

[11] RFM22B/23B ISM Transceiver Module, Datasheet, Hope Microelectronics Co., LTD, V1.0, 70 p.
Saatavissa: http://www.hoperf.com/upload/rf/RFM22B_23B.pdf

[12] SPI Block Guide, Freescale Semiconductor, Inc., Motorol, Inc., V04.01 Document Number S12SPIV4/D, 40 p.
Saatavissa: http://www.engr.colostate.edu/ECE251/S12SPIV4.pdf

[13] Antti Räisänen, Arto Lehto: Radiotekniikan perusteet, 12. Painos, Oy Yliopistokustannus/Otatieto, Helsinki 2007, 287 s.

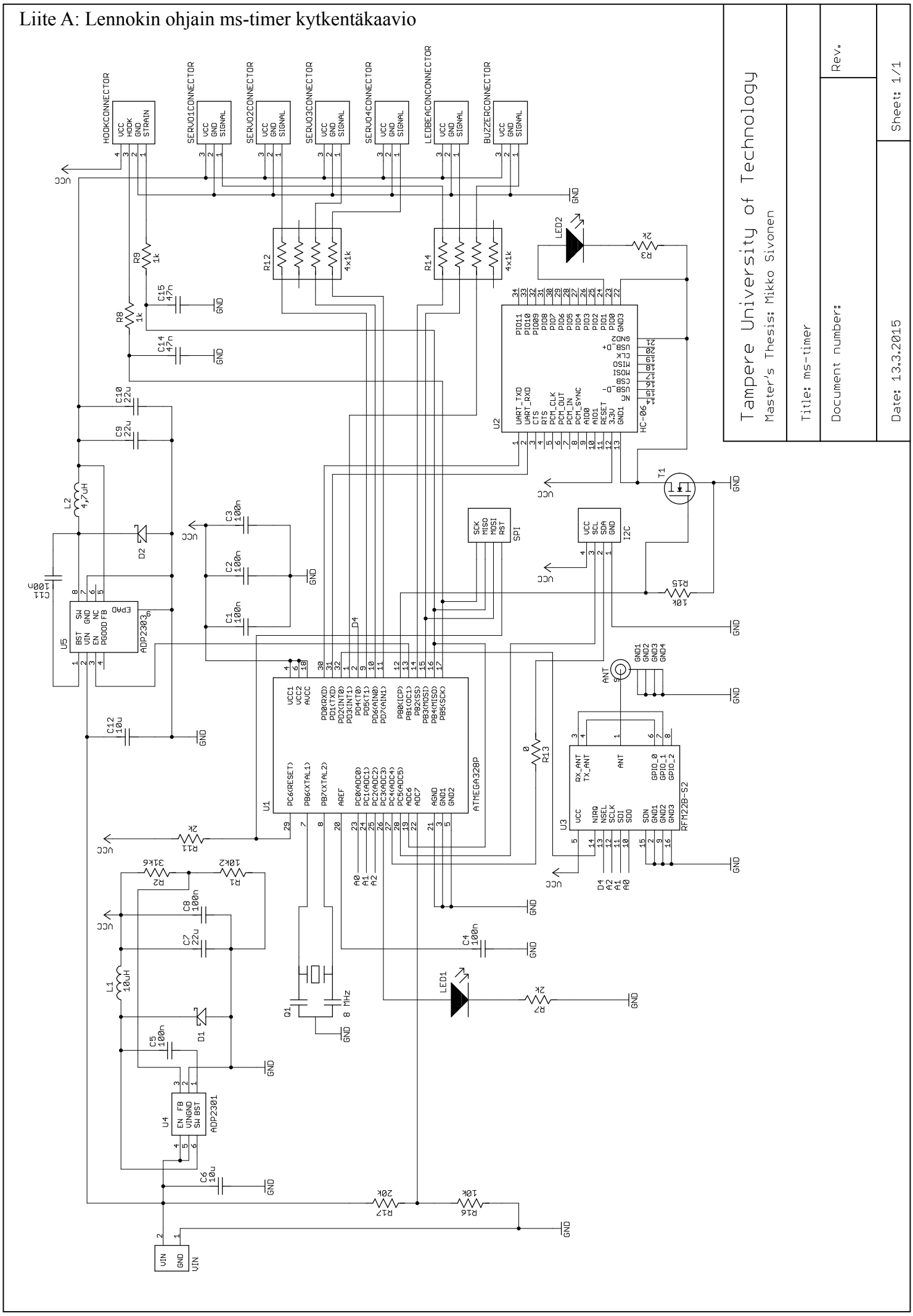
[14] Arduino, Arduino LCC, verkkosivu
Saatavissa: (viitattu 10.2.2015) www.arduino.cc

[15] OpenLRS, Open Source RC System, verkkosivu
Saatavissa: (viitattu 1.2.2015) https://code.google.com/p/openlrs/

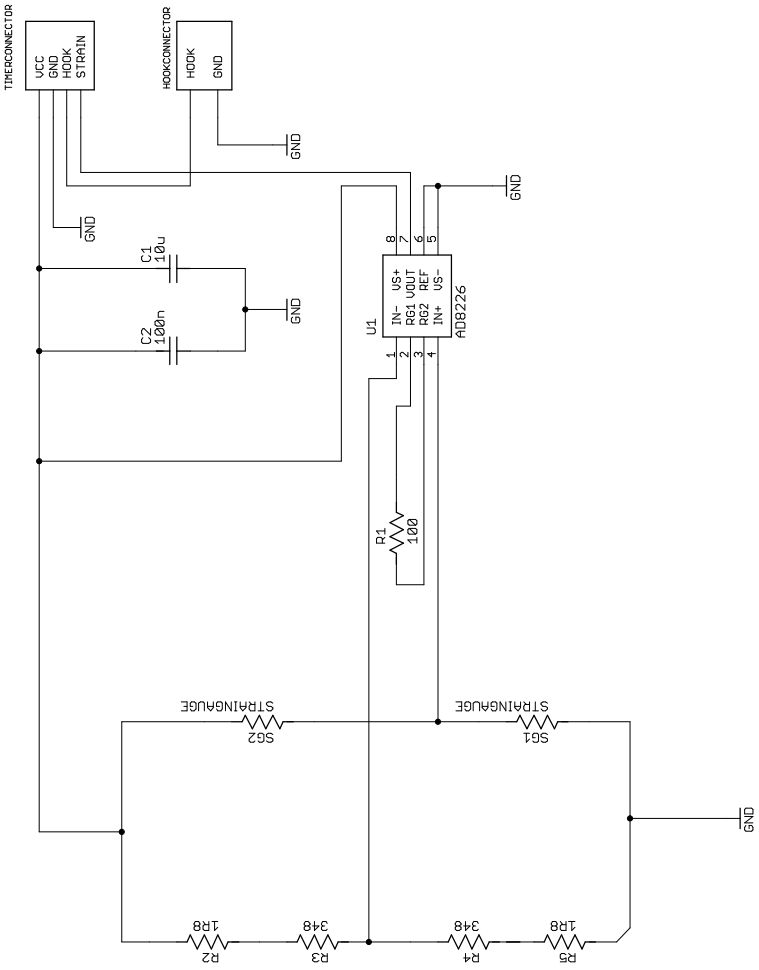
[16] Processing, verkkosivu, Processing Foundation
Saatavissa: (viitattu 15.2.2015) www.processing.org

[17] ControlP5, A GUI (graphical user interface) library for processing, Andreas Schegel, verkkosivu
Saatavissa: (viitattu 15.2.2015) http://www.sojamo.de/libraries/controlP5/

Liite A: Lennokin ohjain ms-timer kytkentäkaavio



Liite B: Hinauskoukun instrumentointivahvistimen kytkentäkaavio



Tampere University of Technology Master's Thesis: Mikko Sivonen	Title: ms-timer hook control board	Rev.
	Document number:	
Date: 13.3.2015		Sheet: 1/1